Touchless Gestural Control of Concatenative Sound Synthesis

Augusto Gabriel Vigliensoni Martin

Music Technology Area
Schulich School of Music
McGill University
Montreal, Canada

August 2011

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Arts.

2011/08/21

Abstract

This thesis presents research on three-dimensional position tracking technologies used to control concatenative sound synthesis and applies the achieved research results to the design of a new immersive interface for musical expression.

The underlying concepts and characteristics of position tracking technologies are reviewed and musical applications using these technologies are surveyed to exemplify their use. Four position tracking systems based on different technologies are empirically compared according to their performance parameters, technical specifications, and practical considerations of use.

Concatenative sound synthesis, a corpus-based synthesis technique grounded on the segmentation, analysis and concatenation of sound units, is discussed. Three implementations of this technique are compared according to the characteristics of the main components involved in the architecture of these systems.

Finally, this thesis introduces *SoundCloud*, an implementation that extends the interaction possibilities of one of the concatenative synthesis systems reviewed, providing a novel visualisation application. SoundCloud allows a musician to perform with a database of sounds distributed in a three-dimensional descriptor space by exploring a performance space with her hands.

Résumé

Ce mémoire de thèse présente une nouvelle interface pour l'expression musicale combinant la synthèse sonore par concaténation et les technologies de captation de mouvements dans l'espace.

Ce travail commence par une présentation des dispositifs de capture de position de type main-libre, en étudiant leur principes de fonctionnement et leur caractéristiques. Des exemples de leur application dans les contextes musicaux sont aussi étudiés. Une attention toute particulière est accordée à quatre systèmes: leurs spécifications techniques ainsi que leurs performances (évaluées par des métriques quantitatives) sont comparées expérimentalement.

Ensuite, la synthèse concaténative est décrite. Cette technique de synthèse sonore consiste à synthéthiser une séquence musicale cible à partir de sons pré-enregistrés, sélectionnés et concaténés en fonction de leur adéquation avec la cible. Trois implémentations de cette technique sont comparées, permettant ainsi d'en choisir une pour notre application.

Enfin, nous décrivons *SoundCloud*, une nouvelle interface qui, en ajoutant une interface visuelle à la méthode de synthèse concaténative, permet d'en étendre les possibilités de contrôle. SoundCloud permet en effet de contrôler la synthése de sons en utilisant des gestes libres des mains pour naviguer au sein d'un espace tridimensionnel de descripteurs des sons d'une base de données.

Acknowledgments

I would like to thank my thesis supervisor, Marcelo Wanderley, for having accepted me in the IDMIL laboratory and given me constant guidance and advice on my research, Fabrice Marandola, whose thoughts about a musical interface to explore a sound corpus in a three-dimensional space inspired this investigation, and Ichiro Fujinaga for his understanding in letting me finish this research while working for him.

I am grateful to the program BecasChile Bicentenario, administered by the Comisión Nacional de Ciencia y Tecnología del Gobierno de Chile, for having granted me funds to do this research, and to Seymour Schulich and the Schulich School of Music for allowing me to be focused on my work and travel to diffuse my findings.

Many thanks also to my colleagues and comrades at the Music Technology area Carolina Brum, Johanna Devanney, Bruno Angeles, Ashley Burgoyne, François Germain, Andrew Hankinson, Jason Hockman, Avrum Hollinger, Joseph Malloch, Bertrand Scherrer, Joseph Thibodeau, and Mark Zadel for the support, encouragement, and wisdom through all stages of this research.

Finally, I wish to thank my family—Justina, Santiago, and Vito—and my true love Antonia, for helping, supporting, and inspiring me every day.

Contents

1 Introduction **1**
1.1 Thesis Overview . . . 2
1.2 Contributions . . . 3

2 Touchless Gestural Control: Sampling the Space **4**
2.1 Position Sensing Technology . . . 4
2.1.1 Characteristics and Performance Parameters . . . 5
2.2 Position Tracker Technology and Systems . . . 7
2.2.1 Capacitive and Electric Field Sensing . . . 7
2.2.2 Ultrasonic Sensing . . . 10
2.2.3 Magnetic Trackers . . . 14
2.2.4 Optical Trackers . . . 20
2.2.5 Computer Vision-based Systems . . . 22
2.2.6 Hybrid Inertial Trackers . . . 24
2.3 Summary . . . 26

3 Chaining Sounds: Concatenative Sound Synthesis **28**
3.1 Origins . . . 29
3.2 Overview . . . 30
3.2.1 Description and characteristics . . . 30
3.2.2 Systems . . . 33
3.3 Interaction and Control with CSS systems . . . 42
3.4 Comparison . . . 45

4 Position Trackers, Experimental Comparison **49**
4.1 Position Tracking Systems 50
4.1.1 Vicon 460 50
4.1.2 Polhemus Liberty 240/8 51
4.1.3 Microsoft Kinect 51
4.1.4 In2Games Gametrak 52
4.2 Experiment design and set up 55
4.2.1 Data Acquisition 58
4.2.2 Data Parsing 61
4.2.3 Data Normalization and Mapping 62
4.2.4 Data Recording 63
4.2.5 Data Processing 63
4.3 Results 65
4.3.1 Reported space 65
4.3.2 Accuracy and precision 71
4.3.3 Data measurement rate 73
4.4 Summary 78

5 Immersed in Sounds: *SoundCloud* **81**
5.1 *SoundCloud* 82
5.2 Design and Implementation 83
5.3 Synthesis Algorithms 87
5.4 Interface and Visualization 87
5.5 Discussion 91
5.6 Summary 93

6 Conclusions: Present and Future **94**
6.1 Conclusions 94
6.2 Future Work 95

List of Figures

2.1 System of coordinates of a moving 3D object 5
2.2 Joel Chadabe performing *Solo* with his capacitive sensing-based system (courtesy J. Chadabe) 9
2.3 Parallax Ping))) Ultrasonic Sensor 11
2.4 Michel Waiswisz's The Hands (courtesy STEIM) 12
2.5 The SoundCatcher 13
2.6 Magnetic tracker accuracy degradation, adapted from (Burdea and Coiffet 2003) 15
2.7 The Virtual Musical Instrument environment (courtesy Dr. A.G.E. Mulder, Infusion Systems Ltd.) 16
2.8 The *Virtual Xylophone* (courtesy T. Mäki-Patola) 17
2.9 The *Virtual Membrane* (courtesy T. Mäki-Patola) 18
2.10 The *Virtual Air Guitar* (courtesy of T. Mäki-Patola) 19
2.11 The *Sensemble* inertial measurement unit (courtesy R. Aylward) 25
2.12 *Celeritas'* virtual sphere (courtesy G. Torre) 26

3.1 CBCS schematic diagram, adapted from (Schwarz et al. 2006) 31
3.2 SoundSpotter GUI 35
3.3 2D exploration of a sound corpus in CataRT 39
3.4 openGL implementation for CataRT visualization 40
3.5 Three rotated views of the same timbre-space using timbreID 42
3.6 The Enlightened Hands 44

4.1 PrimeSensor^{TM} block diagram (Primesense 2011) 53
4.2 In2Games Gametrak 54

4.3 2D grid and crates for 3D measurements 55
4.4 Measurement sequence 56
4.5 Measurement of the same point in space with the four tracking systems . . 57
4.6 Position trackers workflow pipeline 58
4.7 Microsoft Kinect calibration user's pose and tracked points 60
4.8 Gametrak modification 61
4.9 Vicon 460 reported space 66
4.10 Microsoft Kinect reported space 67
4.11 Polhemus 240/8 reported space 69
4.12 In2Games Gametrak reported space 70
4.13 Reported data by the four trackers at the origin [0, 0, 0] 72
4.14 Reported data by the four trackers at [-1; -1; 1] 74
4.15 Reported data by the four trackers at [2; 2; 2] 75
4.16 Update rate for the four position trackers 77
4.17 Arrival time difference of the Polhemus data 78

5.1 SoundCloud implementation schematic diagram 84
5.2 SoundCloud Graphical User Interface 88
5.3 SoundCloud space visualisation 90

List of Tables

3.1 SoundSpotter, CataRT, and timbreID CSS systems summary 48

4.1 Vicon 460 MoCap System with M2 cameras @250Hz technical specifications (Vicon Motion Systems 2002) 51
4.2 Polhemus Liberty 240/8 technical specifications (Polhemus Liberty webpage, accessed May 11, 2011) 52
4.3 PrimeSensor™ technical specifications (Primesense 2011) 53
4.4 Gametrak technical specifications 54

List of Acronyms

DMI	Digital Musical Instrument
EMI	Musical Instrument Digital Interface
2D/3D	Two-dimensional/Three-dimensional
OSC	Open Sound Control
DOF	Degrees-of-Freedom
LED	Light Emitting Diode
IMU	Inertial Measurement Unit
IR	Infrared
HID	Human Interface Device
CSS	Concatenative Sound Synthesis
CBCS	Corpus-based Concatenative Synthesis
MPEG-7	Motion Picture Experts Group's Multimedia Content Description Interface
GUI	Graphical User Interface
VGA	Video Graphics Array
UXGA	Ultra Extended Graphics Array
CMOS	Complementary metal-oxide-semiconductor

Chapter 1

Introduction

Nowadays, it is possible to use one's own sound library as a sonic palette for making music. By segmenting a collection of sounds into small *units*, extracting their acoustic features, and arranging them into a descriptor space, a *unit selection* algorithm can find the closest unit to a *target* sound, and concatenate it to the previous one. This kind of synthesis is called *data-driven concatenative sound synthesis* (Schwarz 2004).

As an extension, we can say that in *user-driven concatenative sound synthesis* we can freely interact with the *units* by concatenating one audio segment after another without using an algorithm for a target sound. This kind of interaction has been implemented—with musically interesting results—mainly through the navigation of a two-dimensional descriptor space in a computer interface, typically by using a mouse and keyboard, or a graphic tablet.

The main goal of my research is to provide musicians a system to compose and perform with a *sound corpus* by exploring a three-dimensional space by means of *non-contact gestures*. A "touchless" interface like this will give simultaneous access to more low- and high-level features, increasing control and improving the expressiveness of this kind of sound synthesis, in an immersive and untethered performance.

1.1 Thesis Overview

This thesis is structured in four main parts: (i) a review of the sensing techniques available for spatial data acquisition, (ii) a comparison of professional and consumer-oriented tracking systems, (iii) an overview of concatenative sound synthesis software suites, and (iv) a prototype design considering the previous insights.

Chapter 2 is an extensive description of sensing techniques for three-dimensional position data acquisition. Musical interfaces, instruments, and controllers using these approaches are surveyed and described. The main objectives of this chapter are to identify the technologies already used for spatial acquisition, and how they have been implemented in interface design.

In Chapter 3, I test and compare four tracking systems that use different technologies: the Polhemus Liberty 8[1] magnetic tracker, the Vicon 460[2] motion capture system, the Microsoft's PS Kinect[3] computer vision-based system, and the In2Games' Gametrak[4], a mechanical tracker. Each device is set up in the same room and conditions, and their accuracy and precision, tracker update rate, and the shape of the reported space is measured and compared.

Chapter 4 explores several software suites for concatenative sound synthesis: *Soundspotter* (Casey 2009), *CataRT* (Schwarz et al. 2006), and *TimbreID* (Brent 2010). Their characteristics in terms of segmentation types, sound descriptors, database handling, concatenation types, mapping flexibility, expandability, real-time ability, and dimensionality reduction are compared.

Chapter 5 presents the development of a prototype to track the position of a performer's hands in three dimensions, the mapping of the acquired gestures to the concatenative sound synthesis engine, and the selection of features for each axis of the descriptor space to create a meaningful interface for musical performance. The goal is to provide a performer with the impression of being immersed in the sound corpus.

[1] Polhemus Liberty 8 Electromagnetic Motion Tracking System, `http://www.polhemus.com/?page=MotionLiberty`, accessed May 21, 2011

[2] Vicon Motion Capture Systems, `http://www.vicon.com/`, accessed May 21, 2011

[3] Microsoft Kinect for Xbox 360, `http://www.xbox.com/en-CA/kinect/`, accessed May 21, 2011

[4] Myers, E. E. 2002. A transducer for detecting the position of a mobile unit. UK Patent Application. GB2373039A

1.2 Contributions

As part of courses taken in the M.A. Music Technology program, I have developed interfaces for gesture acquisition in one and two-dimensional spaces (Vigliensoni and Wanderley 2010), (Vigliensoni 2010). This thesis extrapolates these previous works to three dimensions.

Substantial outcomes of my research are threefold: (i) To provide a comparison of professional and consumer-oriented position trackers. This objective updates previous literature (Burdea and Coiffet 2003), reviews[5], and consolidates information presented only in the form of catalogs or unstructured internet information. (ii) To update previous work on compiling historical interactive musical instruments (Piringer 2001) , especially the free gesture ones. Both updated summaries will be part of the ISIDM database[6]. (iii) To create a touchless interface for exploring, selecting and performing with sonic objects. Such an interface could also be used for more general uses, as in the interactive exploration of a catalog of sound effects in the context of audiovisual post-production, or the navigation of a user's song collection beyond the text-based music browser standard.

[5]CNMAT. Position Sensing Technology and Product Summary. `http://cnmat.berkeley.edu/Position-Sensing`, accessed May 21, 2011

[6]The Working Group on Interactive Systems and Instrument Design in Music (ISIDM) is a web resource that aims at compiling references on new interfaces for musical expression. `http://sensorwiki.org/doku.php/isidm/introduction?s[]=isidm`, accessed May 21, 2011

Chapter 2

Touchless Gestural Control: Sampling the Space

A touchless gestural interface is a type of *alternate controller*, which neither resembles nor is inspired by any acoustic instrument, and falls under the sub-category of *expanded-range controllers*, which require little or only limited physical contact to play them (Mulder 1998). This kind of non-contact musical instrument must be tailored to the performer's position, orientation, and movement. These variables need to be measured in a non-intrusive manner, without restricting the performer's movements to be used in the development of an open-air musical interface. The spatial data acquisition of the position of the performer can be done either by using an *egocentric* system of coordinates, i.e., the movement and position of the limbs are measured in reference to the performer's body, or by using the performance space as an absolute system of coordinates. Also, a proper sampling rate should be considered for creating a smooth discretization of the performer's musical gestures. Thus, knowing in advance what kind of musical gestures will be tracked will be worthy to determine the most adequate ways to measure them.

2.1 Position Sensing Technology

A number of different techniques and technologies can be used to sense and measure a physical quantity. Both, the implementation of these techniques and the results we can get with them for performance can be variable. Because of these differences, selecting the proper sensing technology to fulfill the musical needs is a critical step in the project

development.

2.1.1 Characteristics and Performance Parameters

Beyond the intrinsic limitations of the human body, every single part of it has six *degrees-of-freedom* (DOF). In other words, the body's spatial variables can be specified according to both its position and orientation along any of its three axes. Each one of the positional axes is linked to a rotation axis, where *roll* is the rotation around the x-axis, *pitch* around the y-axis, and *yaw* around the z-axis, as can be seen in Figure 2.1.

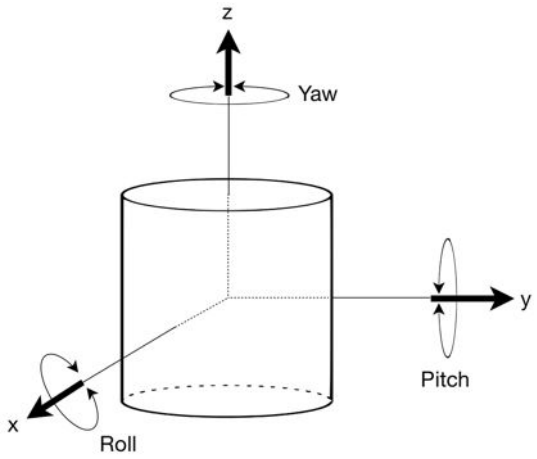


Fig. 2.1 System of coordinates of a moving 3D object

Position trackers share some common characteristics, their *performance parameters*, that describe their behaviour beyond their specific sensing method. These parameters are key-factors in the response of a position tracker and are defined in (Burdea and Coiffet 2003) as:

Accuracy is the difference between the actual value of the object's three-dimensional position and rotation, defined in reference to some absolute standard measurement, and the measured value indicated by the tracker. For any reported value there will be some amount of error due to bias (systematic error) and noise (random error) in the measurement.

Jitter refers to the change over time of the values reported by the tracker when the measured object is stationary, making the reported data change randomly around an

average value. It is also called sensor noise.

Operating range is the space size in which the tracker can perform a reliable measurement of the position and rotation of an object, especially in terms of accuracy and jitter. The acquired data is degraded by the distance between the object and the tracker so the quality of its values is not constant in the sensed space.

Drift represents the undesirable progressive increase of error in the data reported by the tracker. Usually a driftless second tracker is used in companion with the first to reset the reported data.

Latency is the difference in time between the actual position change of an object and the time the tracker needs to detect that change. In complex systems there are several latencies involved: the tracker latency, the communication line latency, and the computer as well as its operating system latency. All those should be considered when calculating the overall system latency.

Tracker update rate constitutes the amount of measurements that the tracker reports per second. In most cases, if several objects are being tracked, this sampling rate is shared among them.

Although the aforementioned parameters can provide a good picture of the response of a position tracker and its peculiar features, in musical practice some of these characteristics are not so relevant when we compare them with other fields of study (such as medical microsurgery where precision and accuracy are critical and, literally, vital). Hence, another suite of parameters more related to musical gesture and music performance can be established to complement the previous ones in the description of a position tracker.

Sensing Method refers to the technique used for tracking the position of one or more points in space. Many techniques can be used, with their own intrinsic constraints and characteristics, so that a review of what is needed to be measured according to musical or performance needs is required to identify the best alternatives.

Sensing Space refers to the contour and shape of the three dimensional space that can be sensed by the position tracker inside the operating range. The performance possibilities will be constrained by this form, so this factor is relevant in the design of a non-contact gestural controller.

DOF stands for the number of degrees of freedom capable of being sensed by the tracker. In the case of position trackers, this item mainly refers to the ability to measure orientation in addition to position.

Number of points refers to the maximum number of points capable to be measured at the same time by the tracker. Also, it is important to know the ability of the tracker to deal with close-located as well as hidden points during performance.

Absolute or relative position refers to the tracker's ability to quantify the actual physical magnitude or only detect a change in the position or orientation of an object.

Occlusion makes reference to how an external object, material, or surface can affect the measurement of the tracker by hiding the target point from the point of view of the tracker.

Portability deals with size, weight, as well as the requirements for setting up the equipment.

Set up and calibration processes and time describes the processes, minimum requirements, and time involved in the process of setting up and calibration of the system.

2.2 Position Tracker Technology and Systems

There have been a large number of position tracker systems developed for diverse applications using different technologies. Because of the aforementioned performance parameters and characteristics, some of these systems are well tailored for musical performance or composition, and others do not.

The following sections will describe how several position tracking technologies used in musical contexts work, and will provide examples of musical interfaces, instruments, and installations developed using these techniques.

2.2.1 Capacitive and Electric Field Sensing

Capacitance is a "quantity describing the charge stored between a set of electrodes" (Paradiso and Gershenfeld 1997) and capacitive sensing is part of a broader class of techniques called *electric field sensing*. In electric field sensing, an electric field can be created using

multiple electrodes. When an external object is introduced in this field, it is possible to measure and compare the amount of current given and transferred by all the electrodes in the system. The value of the electrical field measured by each one the electrodes will depend on the position of the object inside those interrelated fields, and the position of the object can be calculated.

Depending on the amount of electrodes and the setup of the system, three approaches can be proposed (Paradiso and Gershenfeld 1997):

- **Loading mode** makes use of the measurement of the current reduction drawn out from an electrode to ground through a performer's body to know its position in relation to the electrode without any space boundary conditions. In other words, the body can be seen as a virtual ground that steals part of the energy from the system.
- **Transmit Mode** refers to the use of a performer's body to transfer energy from a transmitter electrode to one or multiple receptors instead of sending it to ground. Hence, the body can be seen as a virtual extension of the transmitter.
- **Shunt Mode** is similar to loading mode in the sense that the performer's body draws energy from the system, but this mode allows for more control of the sensing space because boundary conditions can be varied by modifying the position of multiple transmitters and receivers.

Capacitive sensing is not exactly a position tracking technique, however it is a precise proximity sensing technique that is used in many different applications.

Musical Applications

In the context of modern musical instruments, capacitance has been used in the design of several touchless interfaces. The *Theremin*, by Léon Theremin (1919) (Glinsky 2000), is regarded as the first successful electronic musical instrument and it is the first known example of a touchless musical interface.

The Theremin uses a capacitance measurement in loading mode with two antennas, one to control the amplitude and other for the frequency. This instrument takes advantage of the conductivity or our body, so, when a performer approaches one of the antennas, her body increases the capacitance to ground, changing the oscillation frequency or shifting

the phase of a clock signal in an inductor and capacitor system (Paradiso and Gershenfeld 1997). This frequency change is compared with a control oscillator, and the difference between these frequencies is the resulting one. In a similar way, the second antenna is related to a second inductor and capacitor system, allowing the performer to control the gain of a voltage controlled amplifier.

Using similar ideas in terms of performer's interaction and technology, but a different approach in terms of control and synthesis, Joe Chadabe used in his piece *Solo* (1978) a system for interactive composition with two capacitive sensing antennas (Chadabe 1984) that allowed him to control different aspects of precomposed melodies and accompaniment chords. These features could by controlled by means of the proximity of a perfomer's hands to two proximity antennas, allowing him to control the melody speed and note timbre by moving his hands over different zones of the sensed space (Chadabe 1985).

Fig. 2.2 Joel Chadabe performing *Solo* with his capacitive sensing-based system (courtesy J. Chadabe)

The *RadioDrum* (Mathews 1990), a three dimensional baton and gesture sensor also uses capacitive sensing, however it was implemented in a different form. Instead of using a single receiving electrode in loading mode, this device uses two transmitting batons that operate at different frequencies and four receiving electrodes in transmit mode. This technique allows for measuring the proximity of each one of the transmitting batons to each one of the receivers, indicating where the baton is located inside the three-dimensional performance space. The patent of the instrument refers to it as an electronic drum as well as a three dimensional baton and gesture sensor. Thus, this instrument was conceived not for only

discrete triggering of drum sounds but for the continuous control of timbre and sounds by means of moving the batons inside the sensed space.

The Sensor Chair (Paradiso 1994), is another *non-contact musical controller* developed using capacitive sensing . This controller takes the shape of a chair where a performer can sit and control sounds and lighting effects by means of gestures in the air. The designers of the interface attached a copper plate in a chair cushion as a transmitting antenna driven at 70KHz, and four receiving antennas mounted in a frame in front of the chair. When a person was seated in the chair, his body became an extension of the copper plate antenna, and moving his hands inside the frame resulted in a change in the capacitance measured by each one of the receivers. This change in capacitance was analysed in a computer that estimated the position of the hand, and used these values to trigger and modulate the spectral characteristics of sounds selected by foot-switches.

The *Termenova* (Hasan et al. 2002) is a musical interface that combined capacitive and laser-based optical sensing for non-contact gesture tracking in a mostly planar two-dimensional space. This system used the same capacitive principles of the Theremin but provided more solid and longer-range sensing capabilities. It was designed using an electric field sensing circuit in transmit mode that permitted the whole body of the performer to act like a transmitting antenna, making the amplitude of the signal at the receiving side proportional to the distance from the performer to the antenna. The Termenova's visible laser light provided visual feedback to the performer, thus discretizing the performance space. The position of the laser beams could change dynamically with the tuning of the piece. Also, interrupting the laser lights resulted in quantization to the closest diatonic pitch of the played voice. The system also gave the performer the possibility to control the timbre of the sound, and adding delay lines.

2.2.2 Ultrasonic Sensing

Ultrasonic technology uses a sound signal above the range of human hearing to measure the distance between a stationary transmitter and a moving receiver. A short burst of an ultrasonic impulse, typically a pulse train of about 10 square waves at 40KHz frequency, is sent by a emitter and received by a receiver transducer (Bongers 2000). The time elapsed between the emission and the reception of the sound burst is proportional to the distance between both transducers. This method of measurement is know as *time-of-flight* tracking.

With the time-of-flight method it is also possible to detect the reflection of the sound in a non-wired surface or object, making it possible to know where this external object is located in relation to the transducers. Figure 2.3 shows the Ping))) ultrasonic sensor by Parallax. Both transducers, emitter and receiver, are mounted on the same plate.

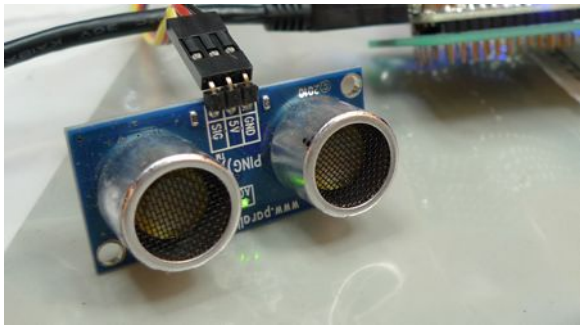


Fig. 2.3 Parallax Ping))) Ultrasonic Sensor

To determine the three-dimensional position of an object in relation to the transmitter or reference point, a *triangulation method* can be used. Given at least three points in space with known coordinates, the unknown spatial position of a fourth one can be calculated by measuring the angles to it from the known points and applying simple mathematical and geometrical relations.

Two topologies are commonly used depending on the specific requirements of a project: one transmitter and several receivers or several transmitters and one receiver (Lima et al. 1996). Furthermore, more complex ultrasonic systems can detect position as well as orientation (Burdea and Coiffet 2003). These systems use the same the triangulation method principle, but use instead a set of three emitters in a plane as well as three receivers in another plane. With this approach nine different distances are measured and the position as well as the angle between the two planes can be calculated.

Although ultrasonic sensing is a very popular non-contact position measuring method because it is cheap, easy to implement, and non lighting-dependant, it has some constraints and drawbacks such as required line of sight, reflections on some type of clothes, low speed of response in multi sensor systems, and susceptible to extraneous noise (Paradiso 1997).

Musical Applications

Ultrasound sensing was implemented in *The Hands* (Waisvisz 1985), gestural interface of the composer and improviser Michel Waisvisz. The first version of this interface used ultrasonic sensing to measure the distance between each hand, by having a transmitter on one hand, a matching receiver in the other, and calculated it with the time of flight method. By means of this, the performer could generate MIDI key-velocity values that were mapped to control separate oscillators in FM synthesis. Figure 2.4 shows the second version of The Hands, where two ultrasonic transducers can be seen attached orthogonally in one of the devices.

Fig. 2.4 Michel Waiswisz's The Hands (courtesy STEIM)

Another early non-contact gesture interface using ultrasonic sensing was used in the piece *Futurity* (1990) (Chabot 1990). The composers' idea was to map a speech sequence into space, as if words were written in the air instead of paper. A vocal sequence was prerecorded in a sampler and its starting point was controlled by the position of a performer in a room. Thus, the performer could playback any part of the sequence by changing her own position in the space. She could also emphasize certain words or syllables, change their order, rearrange the syntax, and change the meaning of the phrases by repeating chunks of the words. The sonar system used in *Futurity* used the 6500 Ultrasonic Ranging device. This sensor was developed by Polaroid Corporation to be used in their autofocus

cameras at the beginning of the 80's and was used by a number of developers, musicians and researchers to create musical instruments and art installations because of its price, ease of use, and availability.

Sonami's *Lady's Glove* (1991) was a very advanced and complex glove controller which gave the performer liberty of movement without any spatial reference, except for her own body. This controller used ultrasonic ranging to measure distance from emitters in the belt, shoe, and opposite arm. Hence, it provided the performer with the ability to use the distance between hands and the left hand height as a control signal to change different sound parameters depending in the mapping and sound synthesis engine (Piringer 2001).

Recently, the *SoundCacther* (Vigliensoni and Wanderley 2010), an open-air gestural controller designed to control a looper and time-freezing patch, was developed using ultrasonic sensing. The goal of the SoundCatcher is to provide singers with the possibility to augment, process, and control their vocal performance for live, rehearsal, composition and recording contexts. This device has a similar spirit to the sonar sensor system used in Futurity, in the sense that it can control the playback head of an audio buffer by means of the performer's gestures. However, in the SoundCatcher the ultrasonic transducers are located at concurrent points on the microphone stand instead of opposite sides of a room. To provide the performer with cues about the sensed space without requiring a computer screen or looking at the microphone stand, two actuators are used to give vibrotactile feedback. Figure 2.5 shows the SoundCatcher with its two pairs of ultrasonic transducers and the motors for providing vibrotactile feedback grabbed by a performer.

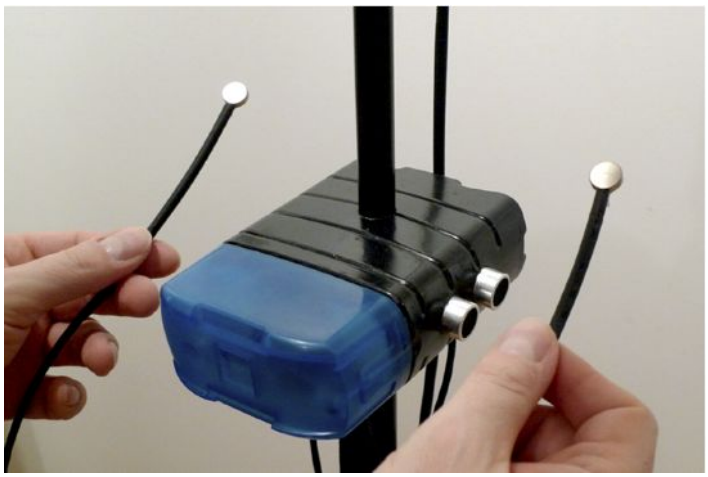


Fig. 2.5 The SoundCatcher

2.2.3 Magnetic Trackers

Magnetic trackers are linear and angular displacement measuring devices based in the quantification of the change in a magnetic field created by a stationary transmitter due to the interference of a moving receiver object (Burdea and Coiffet 2003). The sensing field is created by three orthogonal antennas that are excited sequentially to create three overlapped orthogonal magnetic fields. When a receiver is placed inside this field, the voltages of three little orthogonal coils inside the device change and their values are sampled. Thus, six values, three for position and three for rotation angles in relation to the emitter, are send to an external that calculates the receiver's position and orientation. For the tracking of an external, passive target, such as the hand of a performer, one or more receivers can be attached to the objects or be grabbed by an user.

Magnetic trackers can create AC and DC magnetic fields. While the former suffers from the appearance of eddy current fields due to the presence of surrounding metal surfaces, making the receiver measure distorted values of its position and rotation, the latter are immune to this problem, except with the presence of ferromagnetic metals, which have high magnetic permeability (Burdea and Coiffet 2003). If large, metallic objects are close to the sensed space they need to be removed, otherwise a process of calibration and compensation needs to be done. This calibration process uses a sensor of known characteristics to measure the space at specific locations. As the actual position and nominal measurement values are known in advance, a mapping calculation can be used to compensate the measurement values with the proper ones. However, technical flaws in the calibration process take place when the amount of distortion is large, or the conditions and presence of metallic surfaces in the environment change, making the compensation fails. Hence, it is recommended to use magnetic position trackers in benign environments, free of ferromagnetic and metal surfaces. However, in most of the cases, especially in musical performance contexts, this is not possible because the performer can not control, or know in advance, these variables. Figure 2.6 shows the distortion of the electro-magnetic field due to the presence of a ferromagnetic objects or surface, degrading the accuracy of the measurements.

Musical Applications

Notwithstanding the distortion and calibration issues of electro-magnetic trackers in non-ideal conditions, several musical controllers have been created using this technology. A

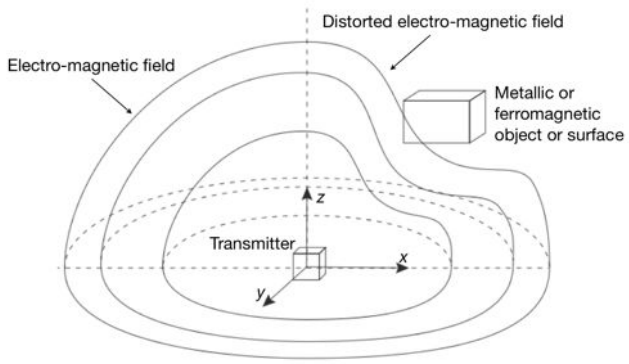


Fig. 2.6 Magnetic tracker accuracy degradation, adapted from (Burdea and Coiffet 2003)

virtual 3D instrument for the control of spatial and spectral sound parameters was designed using the Polhemus Fastrak[1] system (Mulder 1998). The system gave a user the ability to model and control with her hands a virtual instrument projected on a screen by means of non-contact gestures, and used that representation to control a sound engine, as if the sound would be sculpted. The performer's gestures were acquired by using a pair of *CyberGloves*, a wearable device for measuring the shape of the hand and joint data, and the Polhemus Fastrak magnetic tracker with three 6DOF sensors to measure the position and orientation of each hand. These sensing methods acted in complementary ways, one for the absolute quantification of the wrist's position and rotation, and the other for the relative position of the fingers and their overall shape (e.g., "index tip position" or "average finger curvature"). The combination of these sensing methods allowed the system to calculate several hand's features to control different aspects of the virtual object. The author reported that the combination of CyberGloves and the Polhemus Fastrak was good enough to track "all possible hand movements and gestures" (Mulder 1998). However, he did not specify the size of the performance, sensed space. Although visual representation of shapes was not part of the initial project, lastly it was implemented to give the performer with visual feedback to fine-tune her movements. Figure 2.7 shows a diagram with the different devices and blocks used in the system. A representation of the performer's hands manipulating a virtual shape is also shown.

A magnetic-based tracking device called *Pointing Fingers* (Couturier and Arfib 2003)

[1]The Polhemus Fastrak Motion Tracking system `http://www.polhemus.com/?page=motion_fastrak`, accessed June 1, 2011

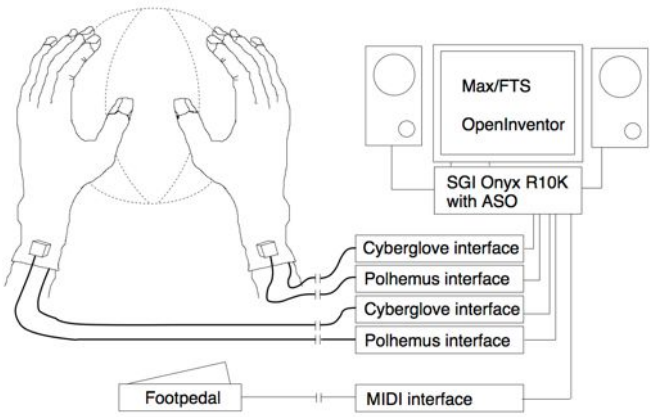


Fig. 2.7 The Virtual Musical Instrument environment (courtesy Dr. A.G.E. Mulder, Infusion Systems Ltd.)

was designed using a multitouch touchscreen-like system. The device used a screen interface where a performer could interact with several graphical objects by touching directly on the screen surface, giving the user the impression that the objects were real. The manipulation of these items allowed the user to control parameters and processes of a Max/MSP emulation of photosonic synthesis (an optical-based sound synthesis), and a scanned synthesis algorithm (a dynamic wavetable-based synthesis that can be created and controlled in real-time). Simultaneous position of up to four fingers was acquired by using using two semi-gloves, two switches per hand, and two *Flock of Birds* 6DOF position and orientation magnetic-based sensors (Ascension Technology Corporation 2011). The system was capable to track the position of the thumb and index finger for each hand, as well as to know if each one of the fingers were touching the screen by means of switches attached to their tips. According to the authors, the *Pointing Fingers* system was successful in providing a tactile way of interaction with a software synthesis patch. However, their system showed some drawbacks. First, the magnetic tracker introduced an estimated latency of about 30ms, too high for musical performance according to the authors. Second, and more important, the CRT computer screen was affected by the magnetic field created by the tracker emitter. The developers changed the computer screen to a LCD screen, but then the screen disturbed the magnetic field of the sensor.

A series of gesture-controlled "virtual reality instruments" using computer vision, magnetic trackers, and data gloves for user input were created to evaluate and analyse their efficiency, learning curve, latency, lack of tactile feedback, and system features (Mäki-Patola

et al. 2004). 3D stereo vision as well as desktop displays provided visual feedback, and vector-based amplitude panning was used for sound spatialization. All the instruments were implemented in a cave-like virtual reality room.

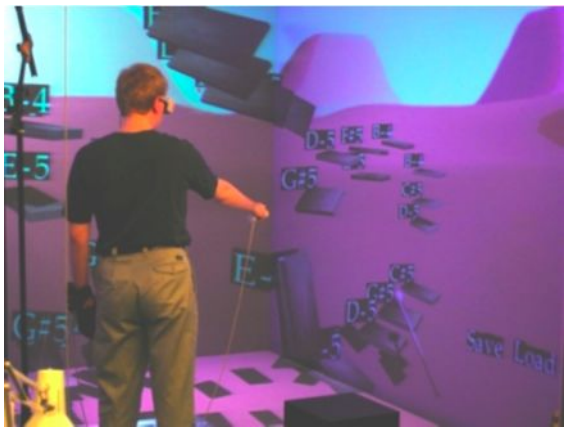


Fig. 2.8 The *Virtual Xylophone* (courtesy T. Mäki-Patola)

The first of the instruments using magnetic tracking was called *Virtual Xylophone*. It consisted in a customizable number of virtual xylophone plates and mallets. These mallets were virtual, i.e., the performer only held two magnetic sensors of an Ascension Technologies' MotionStar system. Although the system did not provide any haptic feedback or resistance, as a real xylophone does, it allowed the performer to try new ways of playing. The plates could be arranged in the performance space at will or at the needs of a specific performance. For example, chords could be created easily by piling three or more plates on top of each other, plates of any size could be created, and they could be struck from both sides (up to down, or down to up), allowing to play them continuously. Moreover, the system allowed for real-time control over amplitude, note, and decay time depending on the striking style. Although the authors considered exciting these new ways of playing, the performance space sensed by the magnetic tracker was rather small, and the paths of playing and kind of movements that the performers used were larger than in real-life conditions because there was not a natural limit on the instrument itself. Furthermore, there was a 60ms latency between the strike of a virtual plate and the reproduction of the xylophone sound, which means that this device can not be used in performance context. Figure 2.8 shows a performer using stereoscopic goggles in the cave-like virtual room holding one of the tracker's magnetic receivers.

A similar approach was used in the design of the *Virtual Membrane* (Mäki-Patola et al. 2004). This instrument was built with the idea of a rectangular drum membrane sound model, whose physical properties could be changed in real-time by the performer. In terms of sensing, the performer's hands were tracked using the same magnetic-based system as the *Virtual Xylophone*, but the sensors were attached to the tip of actual mallets, providing the performer with a more real feel. According to the authors, this system was interesting for users not because of the interaction itself, but because it allowed one to change the physical parameters of a sound model, allowing people to hear results that could not exist in reality. Figure 2.9 shows a performer holding one mallet that has a sensor attached to it. Its positional data is sent to a computer that creates the virtual image that can be seen on the screen.

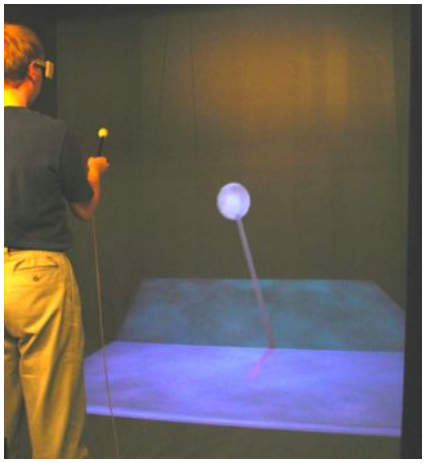


Fig. 2.9 The *Virtual Membrane* (courtesy T. Mäki-Patola)

The last virtual instrument using magnetic sensing for tracking performer's gestures of this series was the *Virtual Air Guitar* (Mäki-Patola et al. 2005). In this interface, the authors developed an entertainment system rather than an expressive instrument. The interface should be capable of tracking the gestures of a performer when playing rock-style electric guitar. An actual guitar was not involved, hence no musical skills were necessary except for the theatrical abilities of a user controlling the virtual guitar purely with gestures. The Virtual Air Guitar was also implemented in a cave-like room with stereoscopic projections and magnetic-based tracking as well as the previous interfaces. The performer's hand position was measured by placing one moving receiver of the position tracker system on each hand. The distance between the two hands was calculated and mapped to the pitch

of an electric guitar sound using a Karplus-Strong model tuned to match a Fender Stratocaster, and routed through a simulated tube amplifier. A scale quantization method was developed to create only notes and chords that can be fitted well together and gesture recognition stage using artificial intelligence was defined and implemented to control different playing techniques such as plucking, strumming, hammer-ons, making slides and vibrato, and so on. Finally, the user could select different play modes for free play, pentatonic scale, soloing and play-along with predefined phrases.

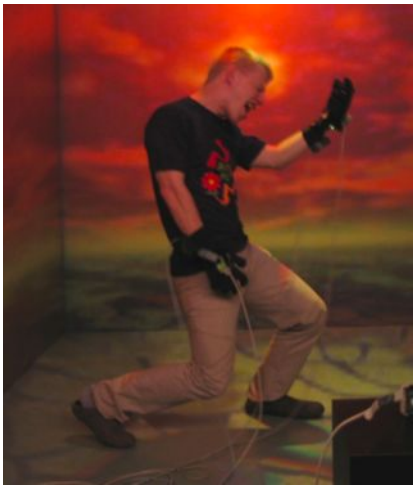


Fig. 2.10 The *Virtual Air Guitar* (courtesy of T. Mäki-Patola)

Among all the aforementioned series of virtual instruments, the Virtual Air Guitar was the most popular interface because it allowed common people to create convincing guitar sounds without any musical skills. Because of this success, it was commissioned for the Heureka Science Centre to be presented for general public, adding more challenges in the design of the user interface. However, the system should be capable of being used by hundreds of people daily, without requiring too much technical setup and support, and easy to use. Thus, a much simpler version using a desktop computer with web-cameras to track coloured gloves was placed in the hands of a user was implemented for a non-invasive tracking. Because commonly computer vision-based systems can track only two dimensions instead of three, and they are slower and less accurate than magnetic trackers or MoCap systems, the new implementation offered less possibilities than the original concept but it allowed to use the same basic architecture (Mäki-Patola et al. 2004). Figure 2.10 shows a user performing with the magnetic tracker version of the Virtual Air Guitar at the virtual

reality room.

2.2.4 Optical Trackers

Optical trackers are camera-based, non-contact position measurement systems that rely on computer vision techniques systems for tracking body parts or objects. The spatial position of these objects is calculated using triangulation, hence more than one camera is needed, but their orientation can not be directly determined since the system only sees points in space. However, a set of three blobs can be defined as a plane and its orientation in space can be calculated. These systems require line of sight, similarly to ultrasonic trackers, but their latency is smaller and their update rate is faster due to the fact that light travels faster than sound. Also, optical sensing does not suffer from the proximity of metallic surfaces, as magnetic trackers do. Additionally, they are capable of tracking markers in larger spaces than their magnetic and ultrasonic counterparts even though their tracking sensitivity is degraded with distance.

Motion capture systems typically use infrared light for tracking beacons attached to a subject or an object whose movements or position are measured. These systems typically generate a stroboscopic infrared light flash that shines on reflective markers located in a user's body or attached to an object. The strobe period is usually synchronized with the period of the opening of the camera's shutter. The lens of the camera collects the light and the camera converts it to video signals. These signals are send to a computer that processes this information and creates a two dimensional representation of a certain marker at a given time (Vicon Motion Systems 2002).

The use of infrared light makes motion capture systems less prone to ambience light fluctuations, but still they need to be calibrated rigorously for avoiding light pollution and reflections. Marker-based optical position measurement can be categorized as *active* or *passive* systems. While the former relies on active, energized, commonly wired beacons; the latter relies on untethered passive markers that reflect light produced by arrays of infrared LEDs placed in the camera or flood by special lighting that can fill the sensed space. Thus, passive sensing optical tracking is commonly used for non-intruding tracking of the human body. Active systems have the advantage that each marker has their own identity code or are time-multiplexed. This feature allows the system to track position and trajectories for adjacent beacons, and markers too close to the line of sight of the

cameras. It also facilitates the reconstruction of the data if the position of a marker is lost temporarily.

Musical Applications

Marker-based optical trackers have been extensively used by artists, composers, and performers in a large amount of musical interfaces and installations.

A system for gestural control of music using the Vicon 8 Motion Capture System was developed and documented in 2003 (Dobrian and Bevilacqua 2003). This research provided to the community of musicians interested in new interfaces for musical expression with detailed documentation about the use, implementation, and implications of this kind of technology for the development of new instruments. The authors of the research developed a Java/C++system called *Motion Capture Music* (MCM) capable of receive, translate and map data acquired with the optical tracker to control synthesis parameters via the MIDI protocol. Although their first prototype only created a direct and linear one-to-one mapping between the spatial coordinate of a specific marker and the value of a MIDI parameter for a certain channel, a second implementation developed in Max (MCMMax) allowed to calculate the velocity and acceleration of any marker in one, two, or three dimensions; the distance and angle between any pair of markers, and provided the user the possibility to select among exponential, logarithmic, reversed, and non-linear mappings, extending the possibilities of the spatial data acquired by the optical motion capture system. This second version also had the ability to analyse and recognize some gestures using principal component analysis. Although the authors do not provide technical details about their results, they posed some thoughts and questions about the aesthetic directions that non-contact virtual instruments could convey, such as how the lack of haptic feedback and no precedence or restrictions in the kind of gestures for performing with touch-less instruments could affect composition and performance, as well as the aesthetics challenges that these circumstances bring. They also envisioned the *management challenge* that large amounts of data can provide, and some issues like portability, set up, and calibration time of optical tracking systems.

2.2.5 Computer Vision-based Systems

Although computer vision-based systems are not precisely 3D position trackers, they have been extensively used to extract and map the position of musicians and artists to musical parameters.

Computer vision-based systems rely on techniques where image information is extracted using video cameras. This data is analysed to extract low-, mid-, and high-level features for a certain image. These systems typically lack of accuracy and precision because they use video cameras running at the video low-frame rate. Hence, latencies are difficult to avoid, making them hard to implement for musical applications. However, meaningful, expressive gestural information starting from a raw image and its combination with other sensor data can be extracted. Some of the techniques for processing video-images are stereo image-based segmentation, colour segmentation, contour detection, connected component algorithms, and image differencing (Von Hardenberg and Bérard 2001).

A four-layer conceptual framework explaining what type of information is acquired and generated for each stage when extracting data using optical marker-less systems has been suggested (Camurri et al. 2004)

- **Layer 1** acquires and analyses physical signals using background subtraction, motion detection, and motion tracking techniques (using colour tracking or optical flow based feature tracking).
- **Layer 2** calculates low-level motion features using computer vision techniques and statistical measuring.
- **Layer 3** computes mid-level features and maps, segmenting motion, trajectories, and gestures in absolute and relative durations.
- **Layer 4** recognizes and classifies the segmented gestures into high-level features such as basic emotions using artificial intelligence techniques and tools.

Systems

The *EyesWeb* (Camurri et al. 2000) software is a marker-less computer vision-based system that uses multiple cameras to extract motion information and perform real-time analysis of body movement and gestures and it also allows for the acquisition of body movement

data through the use of wireless body sensors (e.g., accelerometers). The EyesWeb system is capable to interact with the external world by extracting and analyzing gestural emotion from the performers or users and react to it. Thus, the system has been used for generating sounds and controlling sound synthesis parameters, visual media, lighting, and actuators in the context of dance, theatre, live electronics, or museum and art installations.

Other camera-based systems are the *Very Nervous System* (Rokeby 2010), *BigEye* (Demeyer 1996), and *Cyclops* (Singer 2010). The BigEye software software is capable of react to up to 16 different objects entering predefined 'hot zones' in the visual field according to their colour, brightness, and size. Three states are possible for each hot zone: an object entering the zone, leaving it, or moving inside it. Each one of these states can trigger different actions according to simple mappings or complex rules defined with conditions for different cases in a scheduler (Demeyer 1996).

The Very Nervous System was developed mainly for being used in sound installations. It can use one or many video cameras to observe a space, mapping the video image onto a user definable grid of up to 240 regions where each sector can be defined as a hot zone. The system works by measuring changes in the intensity of light for each zone between current and past video frames. It has two basic modes: motion analysis and presence. While the former compares a current frame with the previous one, the latter compare the present frame with a pre-recorded background image (Winkler 1997).

Musical Applications

The Very Nervous System has been used with two video cameras in the interactive installations *MAP1* and *MAP2* (Paine 2004). The author developed a 3D space which could be entered and played by means of open-air gestures. The sensed space was broken into four horizontal zones, each one of them assigned to three different instrument that responded to changes in the amount of light in their specific zone (i.e., the activity in that area was increased), creating changes in the instrumentation activity for that sector. Thus, there was no pre-recorded material and all music was composed in real-time according to the speed, direction and position of people inside the installation, creating overlapped sound textures .

The piece piece for live electronics and violin *Transformation* uses a computer vision application to control concatenative sound synthesis and sound spatialisation by means of

the position of a violinist on stage (Jensenius and Johnson 2010). Details of this implementation is given on subsection 3.3.

2.2.6 Hybrid Inertial Trackers

Inertial trackers are self-contained position sensing devices capable of measure the relative change in position and orientation of an moving object without relying on external points of reference. Three orthogonal gyroscopes are used to measure the angular velocity on each axis. The reported values by the gyroscopes are integrated over time to calculate the orientation of the object. At the same time, a set of three accelerometers coupled with the gyroscopes measure accelerations with the object itself as a reference point. The reported data from the accelerometers is integrated over time twice and the position of the object is calculated according to a reference point defined during calibration.

Advantages of inertial trackers are the theoretical unlimited sensing space because of its sourceless operations, no line of sight constraints, and very low sensor error, resulting in low latency because no much filtering is needed. The main drawback of inertial systems is their accumulative calculation error over time, i.e., drift. In order to overcome this problem, another technology can be used to reset or update the system. Data from different sources can be combined to make the measurement more reliable. Thus, systems that use more than one technology for refining the tracking measurement are known as *hybrid trackers* (Burdea and Coiffet 2003).

Musical Applications

Several musical interfaces are described in the literature using inertial measurement units (IMUs). The *AirDrum* (Chabot 1990) is a device that used accelerometers attached to sticks for tracking vertical, horizontal, and rotational motion. While the performer held one stick in one hand, a second stick was attached to her foot. The data acquired by these sensors was combined with ultrasonic sensing data to improve the accuracy and precision in the aforementioned piece *Futurity* (see section 2.2.2).

The *Sensemble* (Aylward and Paradiso 2006) is a wireless multi-sensor system for interactive dancing. This system consisted on sensor modules based on 3-axis gyroscopes and accelerometers that measured the motion in the wrists and ankles of dancers. By using inertial tracking they solved problems of occlusion and the use of wireless transmission

allow them to measure the dancers body motion unobstructedly. However, because of the large amount of data generated by many performers at the same time, the authors used statistical methods for extracting meaningful information, making the motion features vary slowly and increasing the latency. Hence, their system was not appropriate for triggering sound or controlling sudden events, but for commanding the overall music progression and adding ornamentation. Figure 2.11 shows an IMU node fixed to a performer's wrist.

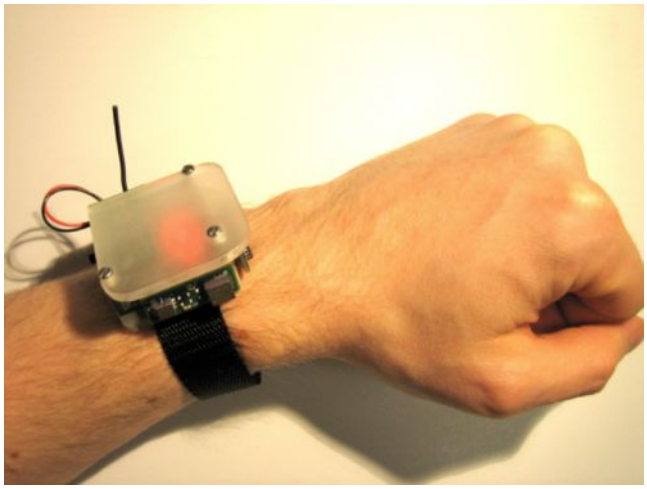


Fig. 2.11 The *Sensemble* inertial measurement unit (courtesy R. Aylward)

The *Celeritas* (Torre et al. 2007) is a wireless 6DOF IMU system similar to the *Sensemble* but developed to be used for solo or group dance performers and with two dual axis magnetometers that increased the accuracy of the system. In their system, a dancer used 5 units to create a virtual sphere centred in her chest with a radius determined by the tips of her fingers and toes. This idea of giving the performers an explicit performance space is interesting because they can limit their movements to that specific space. The performance space of Celeritas can be seen in Figure 2.12.

A modern interface using inertial properties sensing is the *T-Stick* (Malloch 2008). This controller is capable to report acceleration at both ends of the device by means of two linear three-axes accelerometers integrated circuits. Although the system does not calculate or extract the relative T-Stick position, it measures the values of the tilt, or inclination, of the interface with respect to ground, and use these values to control user-definable parameters in a sound synthesis engine.

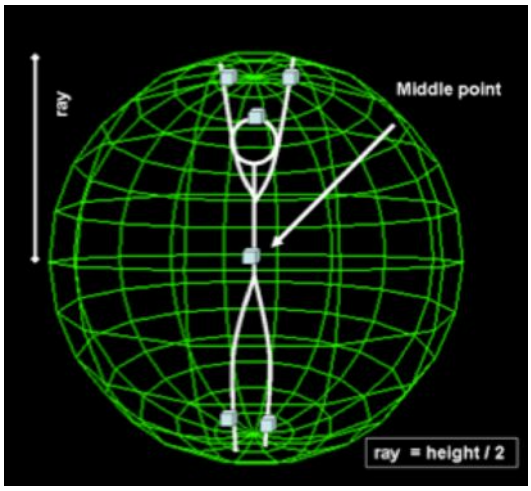


Fig. 2.12 *Celeritas'* virtual sphere (courtesy G. Torre)

2.3 Summary

This chapter described some of the characteristics that touchless gestural interfaces have, in the context of the development of new musical instruments. The performance parameters of systems to unobtrusively track and acquire the gestures of a performer were defined and explained, and the most common technologies for non-contact spatial tracking were reviewed. The use of these techniques in diverse musical contexts was exemplified by describing some of the musical interfaces developed using devices based on those technologies.

It can be observed that there not exist a best, unique position tracking system to fulfill the requirements for every context. The same parameter can be measured by means of several techniques, and diverse systems implementing these techniques can have different characteristics and performances parameters. These differences make that one system could work good for one particular case, and do not work in other. Thus, different designs and approaches can address, or not, the requirements of a specific musical performance. In particular, when designing interfaces for musical expression, a set of other practical considerations that are inherent to the context of live-music appear: the system should be portable but robust, easy to set up and calibrate, cheap or reparable, immune to environmental on-stage conditions, and so on. Still with these constraints, the system must allow to acquire accurately the performer's musical gestures, and to let the user to map those gestures with flexibility to the synthesis engine.

Finally, the performance parameters as well as other technical features of devices for spatial tracking must be considered in the implementation of a system for musical performance, but the specific musical needs of a piece or system and the gestures it requires should govern the design of a new interface for musical expression.

Chapter 3

Chaining Sounds: Concatenative Sound Synthesis

We have entered a technological era that allows us to search for music and sounds beyond one's own music collection. Digital music services, content resolvers, music aggregators and media-sharing websites provide us with real-time access to large amounts of sonic content. Sonic material from this massive repositories can be used to create rich, heterogenous sound corpus to produce new sonic textures by means of *concatenative sound synthesis* techniques.

Concatenative sound synthesis techniques allow a user to segment a collection of sounds into small *units*, analyze their *sonic identity* by means of the extraction of their acoustic and perceptual features using descriptors, and arrange the sound units into a multi-dimensional descriptor space according to their values. By means of the manual or automatic selection of sound units from the database and their concatenation, we can synthesize new sonic textures.

In this chapter, we present a general overview of concatenative sound synthesis and a comparison of the characteristics of three real-time, open-source implementations of this kind of sound synthesis: *SoundSpotter*[1], *CataRT*[2], and *timbreID*[3]. The study and comparison of the intrinsic characteristics, features, and performance possibilities of these systems will inform our research on how to augment and enhance the gestural control of concatenative sound synthesis.

[1] `http://soundspotter.org/`
[2] `http://imtr.ircam.fr/imtr/CataRT`
[3] `http://williambrent.conflations.com/pages/research.html`

3.1 Origins

Although some authors have envisioned concatenative sound synthesis (CSS) as "a new approach to creating musical streams by selecting and concatenating source segments from a large audio database using methods from music information retrieval" (Casey 2009) or as "a promising method of musical sound synthesis" (Schwarz 2006), in fact this technique has been used for years in speech synthesis in what is named diphone synthesis (Rodet et al. 1988).

Also, somehow related musical examples can be traced 60 years before, when Pierre Schaeffer worked with heterogeneous corpuses of sounds, subjectively analyzed, manually selected, segmented and catenated, for the creation of new sonic textures in what is called Musique Concrète. Since then, many composers such as Karlheinz Stockhausen in"Étude des 1000 collantes" (1952), John Cage in "Williams Mix" (1952), Iannis Xenakis in "Analogique A et B" (1958–59), John Oswald in his series of albums "Plunderphonics" (1988–89), "Plexure" (1993), and "69plunderphonics96" (2000); and Bob Sturm in "Concatenative variations of a passage by Mahler" (2005), among many others, have taken similar approaches for creating music (Schwarz 2006).

CSS has two main origins. On the one hand, it owes to previous research on speech synthesis, and on the other hand to the conceptual idea behind granular synthesis.

The first concatenative synthesis techniques were developed as part of text-to-speech systems. In these systems, a user types arbitrary text on a computer terminal, words in the text are segmented into elementary units, these units are analyzed and transcribed to phonemes using linguistic analysis, and then the phonemes are used as control parameters for waveform synthesis or linked to sampled speech sounds in a database by means of a unit selection algorithm. To synthesize the output signal, these sound units are then concatenated. These concatenative techniques were later on applied to synthesize singing voice, and more recently to sound synthesis (Schwarz 2004).

In granular synthesis (Roads 2001), sound is produced by taking short snippets of sound called *grains*. These grains, microacoustic events with a duration between 1 and 100ms, can be played back with different pitch, volume, and envelope, and can be used as building blocks for the design of new sound objects. By layering thousands of grains with variable length and position, in what is called a *cloud* of grains, sound atmospheres of different characteristics can be created. Granular synthesis relates to CSS in the sense that it works

with minute sound units that can be catenated one after another to create new sound textures.

3.2 Overview

CSS uses a large amount of heterogeneous, previously segmented sounds units stored in a database, to synthesize new sounds. Descriptor values representing the acoustic and perceptual features for each sound unit in the *sound corpus*, their sonic identity, are extracted and analyzed. If a matching algorithm, symbolic score, set of rules, or another sound is used to search for the closest-distance unit in the sound corpus according to an arbitrary user-defined set of descriptors, the process is called *data-driven* CSS (Schwarz 2004). If the sound units are manually selected from the system's database by a user according to any arbitrary set of descriptors the process is called *user-driven* CSS. The generic name, which refers to both cases, data-driven as well as user-driven CSS is *corpus-based concatenative synthesis* (CBCS) (Schwarz et al. 2006). In CBCS, sound units from a large database of sonic content are analyzed, selected and catenated one after another, creating sequences of sounds that could resemble in any desired degree a target sound, or produce novel sound textures.

3.2.1 Description and characteristics

Concatenative sound synthesis systems comprise at least three main stages. First, an *analysis* stage where sounds in the database as well as the target sound (if any) are segmented and analyzed according to some descriptors. Second, a *selection* stage where an unique sound unit in the database is chosen according to the measurement of the closest geometric distance between chosen user-driven or data-driven multi-dimensional descriptor vectors, and the descriptor vectors for sound units in the database. Finally, a *synthesis* phase where the selected units are concatenated one after another. Figure 3.1 shows a schematic diagram of a CBCS system adapted from (Schwarz et al. 2006). Processes involved in the analysis stage are plotted light blue, selection processes are green, and synthesis processes are yellow.

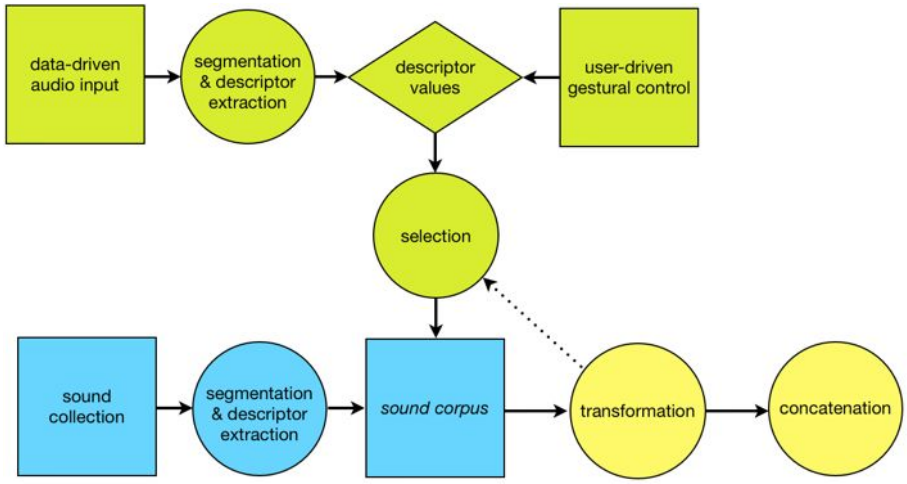


Fig. 3.1 CBCS schematic diagram, adapted from (Schwarz et al. 2006)

Analysis

The analysis stage performs the segmentation, extraction, and analysis of features that describe the identity and intrinsic characteristics of sounds in a *sound corpus*.

The segmentation process implies splitting the signal over time, allowing the system to extract descriptor values for each one of these sound units, and store these values in a feature vector linked to each one of units. This stage is one of the core issues in CSS because it defines the corpus' sound units that will be used for selection and concatenation.

One of the main attributes in the segmentation process is the window length. This variable can control the trade-off between the density of spectral information and the temporal resolution. It has been pointed as "one of the most important parameters for controlling the relationship between the identity of the target signal and the identity of the source in the matching process" (Casey 2009). While longer-windows preserve the identity of sounds stored in the database, shorter-windows allow to maintain the identity of the target audio signal (if any) in the resulting output signal. This feature is particularly relevant in data-driven CSS because target and source sounds are used, and the user can control the *sound identity ratio* between them. Using smaller windows for the segmentation will create smaller audio units with a more accurate description of their content, but offering less musical identity. Larger windows in general will help to preserve the musical identity of the units, but the descriptor values will be averaged over a longer window, so they will

be less accurate. Another advantage of using variable-length windows is that the number of units in the database can be lowered, diminishing the amount of computation needed to select an unit from the database, allowing faster matchings (Casey 2009).

To store, compare and retrieve sounds from the database according to user-driven actions or data-driven schemes, the acoustic (signal, spectral and harmonic) and perceptual characteristics of sounds in the sound corpus are extracted. These values represent the *sound identity* of a given sonic object, and can be expressed through a set of descriptors, "values that describe a certain quality of a sound, which can evolve over time or be constant" (Schwarz 2004). These descriptors can be grouped into three types: *low-level* descriptors, which use signal analysis methods to extract information directly from the source sounds, *perceptual* descriptors, which are derived from values from low-level descriptors but are transformed using music perception knowledge to match how humans perceive sound, and *high-level* descriptors that can be manually assigned by the user to express some categorical or subjective attributes for the sounds units, such as the instrument class, and its "glassiness value" or "anxiousness level", for instance (Schwarz 2007). According to their values, the descriptors can also be seen as boolean (with an unique value that express a membership to certain class, for example), static (with a constant value), or dynamic (with values that change over time). For dynamic descriptors, statistical approaches are commonly implemented in order to compare sound units with values that change over time (Schwarz 2004).

Selection

Descriptors represent different attributes of the sound units, so their scales are frequently different. Therefore, to perform searches in a multi-dimensional descriptor space, a data normalization process is commonly implemented beforehand. In data-driven CSS, the real-time selection of sound units is performed by a process that searches for the geometric closest distance between the multi-dimensional descriptor values of sound units in the corpus and a target sound. In user-driven CSS this process of selection is less complex because the user provides, implicit or explicitly, values for a certain set of descriptors, and the closest units in the sound corpus are chosen.

Most CSS systems allow to search for sound units using a multi-dimensional k-nearest neighbour strategy. This method allows the user to choose the radius k of the search,

giving the performer control over the precision of the selection process. Some systems also provides the possibility to weight descriptors to perform a weighted search.

Synthesis

In order to give more musicality to the resulting output by not choosing the same units again and again for an identical input, some systems implement methods to transform or modulate the selected units over time using signal processing methods. Other systems implement methods of short-term and long-term memory to the selection process in order to not repeat units over a certain period of time. This feature produces a change in the selection algorithm according to previous decisions as can be seen in Figure 3.1. The dotted line between the transformation and the selection processes denotes this behaviour, where the output can exert some control in the selection process.

The concatenation of the selected sound units is the last process of a CBCS system. Current real-time implementations allow simple crossfading of sound units and not consider the catenation distance between units. To overcome this problem, most systems provide sonic transformations to create smoother transitions (Schwarz et al. 2008).

3.2.2 Systems

Several CSS systems have been developed in the last 10 years for music research, performance, and creation. The characteristics and features of these systems are diverse, and have been extensively surveyed and compared in (Schwarz 2006) and (Sturm 2006). Among these systems, in this thesis we present and review *SoundSpotter*, *CataRT*, and *TimbreID*, CSS systems with real-time capabilities, a repertoire of musical pieces, and open-source code available.

SoundSpotter

SoundSpotter was designed as a deterministic, real-time, computer music instrument (Casey 2009). It is written in C++ and has external objects for Max/MSP and PureData. The application allows a user to load a sound corpus in memory (bounced as a single audio file) and feed the system with a pre-recorded or real-time, live audio target signal. Descriptor values for audio features in the sound database are extracted off-line, and features from the target sound are calculated *on-the-fly*. When the system is enabled, it selects audio

segments from the database that match the target signal by calculating the closest distance between the descriptors of both audio segments. Hence, Soundspotter is a data-driven CSS system only.

Soundspotter's analysis stage can perform the following types of segmentation of the audio content: *periodic*, *inter-onset* , and *tempo-based* segmentations (Casey 2009). While periodic windowing creates fixed-length segments, inter-onset segmentation automatically changes the length of the window according to on-sets in the signal, processing musical content of different nature adaptively. Therefore, sound sources with more percussive, on-set oriented elements are processed using shorter windows, and texture-oriented, less percurssive sounds are segmented using longer windows. By this means, the system adapts itself to the musical content which is processed, saving computation cycles, performing faster searches, and producing more musical-informed segmentations. Finally, tempo-based is similar to inter-onset segmentation, but the intervals are calculated in SoundSpotter by tracking the tempo of the target sound.

To describe the source sounds and the live target signal, Soundspotter's descriptor extraction analysis stage is based on log-frequency cepstral coefficients (LFCC) (Casey and Grierson 2007). These values represent the short-term power spectrum of a sound, and are similar to mel-frequency cepstral coefficients (MFCC). As a musical instrument, SoundSpotter implements the full range of vector values, 89 coefficients in total instead of the 12 or 20 values commonly used for MFCC in speech recognition systems, because we can not know in advance which coefficients will provide a better match between the target signal and sounds in the database. An advantage of using a cepstral representation is that the harmonic content of a given signal, represented by periodic components in the spectrum, is separated from timbral components due to formants (Casey 2009). Hence, this representation allows to separate, according to the selection of different set of coefficients, the timbral and pitch content in the signal. This feature lets the user to opt for a more pitch, timbral-centred, or any point in between comparison of the source sound units and the target signal (Casey and Grierson 2007).

SoundSpotter uses a *sequence matching* algorithm to select the best matching unit in the database according to the target sound. This algorithm is based on the cross correlation computing for all selected features of the audio segments. The target audio segment slides across units of the same window length in the database and calculates the cross correlation for each pair of segments, where larger cross correlation values indicate more similarity.

The largest value is chosen as the matching unit (Casey 2009).

In the synthesis stage, SoundSpotter implements several *memory-related* methods to control the way the closest matching units are selected. The *memory loss* method controls the range of memory used for the selection, *frameQueue* removes from memory sound units that have been already played for a certain time, and *feedback matching* allows the user to use a proportion of the resulting output signal to feed back the system, thus creating smoother transitions.

Figure 3.2 shows the SoundSpotter GUI in Max/MSP. The system allows access to the range of LFCC dimensions, the size of the window, the radius of k for a knn algorithm, and time values for the memory loss features.

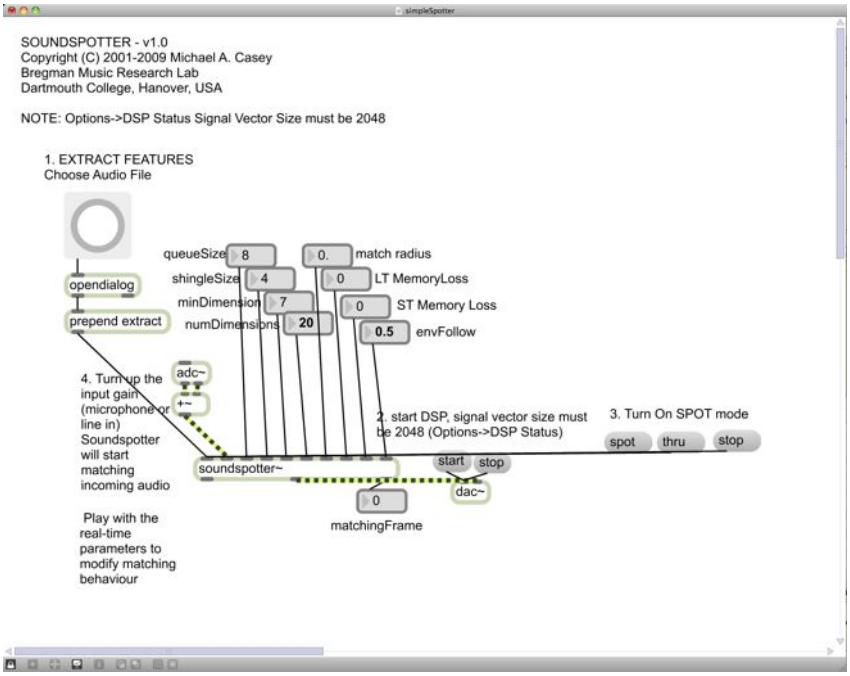


Fig. 3.2 SoundSpotter GUI

The following is a summary of the key elements in the design and implementation of SoundSpotter:

- Target Sound: the process that drives the SoundSpotter system is a live audio target signal

- Determinism: for a given target sound, the system should select from the sound database the same sound.
- System Response: the response of the system to a given target signal should be "instantaneous".
- Database Structure: the sound database is organized as a single, large file of sounds.
- Real-Time Control: the user has real-time control of the process of matching the target sound with the sound database.
- Perceptual Features: to find the closest match between the target sound and a sound in the database, the system uses perceptual features as sound descriptors.
- Segmentation: length can be automatically changed *on-the-fly* according to the musical content.
- k-Selection: Real-time control of the matching distance between the target and the sounds in the database. Sometimes, more interesting results can be achieved by not selecting the perceptually closest sound unit.
- Memory Loss: this feature allows to remove temporarily from the search engine source segments that have been played, giving the user the possibility to explore part or the whole sound corpus without repeating sound segments.

These attributes are a good declaration of principles for the design of SoundSpotter as a computer music instrument. However, their implementation flaws at least in the following aspects:

1. The system is not "instantaneous", nor close to instantaneous. All signal processing in SoundSpotter works with a hard-coded buffer value of 2048 samples. Hence, the sound card buffer size must be set up with the same value, providing a constant latency of 46.44ms at 44.1KHz.
2. For efficiency, the system only works with a single audio file. This issue can be problematic and can slow the creative process when testing sounds to create new sound corpuses.

3. The real-time interaction possibilities offered by SoundSpotter are powerful, but restricted and non-intuitive. Only eight user-chosen variables control all analysis, selection, and synthesis processes of the system. These variables are high-level, they can be accessed only as numerical values in the GUI and their effect at first glance in the resulting output is not always clear.

4. The externals for Max/MSP and PureData, as well as the source code are not well documented.

5. SoundSpotter only works as a data-driven CSS system.

CataRT

CataRT is a real-time system for the navigation of a multi-dimensional descriptor space of a sound corpus in a process called *corpus-based concatenative sound synthesis* (Schwarz 2007). The system is based on a collection of Max/MSP patches using the FTM [4], Gabor, and MnM libraries (Schwarz et al. 2008). These libraries extend the possibilities and features of Max/MSP in terms of matrix processing, creation of data structures, mapping, and processing of signals at arbitrary rates.

CataRT plays sound units selected from a sound corpus according to a target descriptor extracted from a live-signal or pre-recorded sound file, or to an user-defined set of descriptors. To select the sound unit, CataRT calculates the distance between the target and source multi-dimensional descriptors and selects the closest unit.

The system's architecture is organized in modules. These modules handle all aspects of analysis, selection, user interaction, and catenation of sound units, and some of them can be instantiated several times. The main modules are `catart.import`, `catart.analyzer`, `catart.data`, `catart.init`, `catart.lcd`, `catart.synthesis~`, and `catart.select` (Schwarz et al. 2006).

Segmentation and descriptor calculation can be done in an external, third-party application, and loaded afterwards in CataRT. However, the application offers *on-the-fly* capabilities to perform these processes in a single-sound basis or in batch-process. In terms of segmentation, CataRT offers the following modes: *chop*, which segments the signal into arbitrary, equal-sized units; *split*, which divides a sound source into a user-defined number

[4] `http://ftm.ircam.fr/index.php/Main_Page`

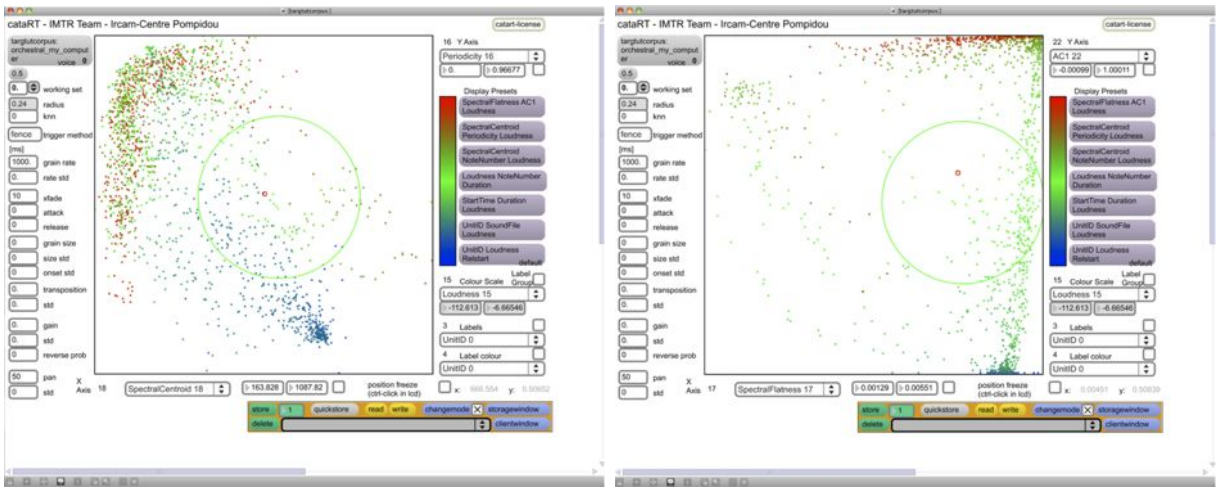
of same-length audio segments; *silence segmentation*, which uses a threshold and minimum and maximum durations to split a sound; *yin note segmentation*, which segments by pitch changes; *none*, for not segmenting sounds, as in the case of percussive sonic object; and importing pre-segmented and analyzed labels for sound units from third-party applications (Schwarz 2011). These segmentation possibilities cover a wide range of applications, but do not offer an adaptive method according to the musical content.

CataRT uses 25 descriptors to express each sound unit's low- and high-level unique identity. The first 13 descriptors are directly extracted from the unit's sound file attributes (such as *UnitID*, *RelID*, *SoundFile*, *StartSample*, *SampleSize*, *StartTime*, *Duration*, *RelStart*, and *Shift*) or assigned from high-level, user annotated attributes (*UnitType*, *SoundSet*, *WorkingSet*, *Active*). The second group of 12 descriptors is extracted using some of the MPEG-7 ISO/IEC standard's signal, perceptual, spectral, and harmonic descriptors (Schwarz et al. 2008). These descriptors are the *fundamental frequency* (and *MIDI Note Number*), *aperiodicity*, *loudness*, *spectral centroid*, *sharpness*, *spectral flatness*, *high-frequency energy*, *mid-frequency energy*, *high-frequency content*, *first order autocorrelation coefficient*, and *energy*. Details for each descriptor can be found in (Schwarz 2004).

A set of statistical values (mean, variance, slope, min, max, and range) is calculated for the time-evolving raw descriptors for each sound unit. These *characteristic values* (Schwarz 2004) express the evolution in time of the descriptors for a given sound unit, and allow for a faster comparison and retrieval according to a target descriptor value.

To determine the selected unit according to a given target descriptor vector, CataRT searches for the closest unit, in a geometrical sense, in the multi-dimensional descriptor space. To calculates the shortest path, the Mahalanobis distance is computed for all units and the target position(Schwarz et al. 2008). Moreover, the user can provides a value r to randomly select sound units inside this radius r, or among the k closest units to the target vector.

Once each sound unit is selected, the attack, release and time of a crossfade is selected to smooth the transition between each grain. Furthermore, a probabilistic value for each played back sound unit's transposition, pan, gain, and reverse probability is user-defined. CataRT also offers *trigger modes* to specify how and when the grains can be played back. These modes, *bow*, *fence*, *beat*, *chain*, *quant*, *seq*, and *cont*, allow the user to play back units each time the mouse is moved, when a different unit becomes the closer one, synched to selection to a metronome or sequencer, when a unit has finished playing, or to play back



(a) Spectral Centroid vs. Periodicity

(b) Spectral Flatness vs. First Order Autocorrelation Coefficient

Fig. 3.3 2D exploration of a sound corpus in CataRT

only continuous units.

The `catart.lcd` module allows the user to interact with the sound units directly in a 2D space by means of selecting a set of 2 descriptors, each one for each axis. The colour of the units can also be linked to a third descriptor. The user can explore this space by moving the mouse and dragging a radius of selection. Figure 3.3 shows 2D space representations of a sound corpus populated with more than 2000 sound units, but with a different set of descriptors for the x and y axis, and colour. Figure 3.3(a) shows the sound units distributed in the 2D space according to their spectral centroid and periodicity. Figure 3.3(b) shows the 2D space populated with the units but using the spectral flatness and the first order autocorrelation coefficient as the x and y axis, respectively. In both figures, the colour is linked to the loudness. The green radius shows the range where a sound unit can be selected.

Other representations to enhance the visualisation and interaction with the CataRT's descriptor space have been developed using its open and modular architecture. Figure 3.4 shows Van Der Wee's OpenGL interface [5] to visualize and browse a 3D representation of a sound corpus. The Figure shows the three axis as well as the sound units distributed in the space according to their descriptor values. This descriptor space can be rotated and

[5] OpenGL interface for CataRT available in the CataRT mailing list at `http://listes.ircam.fr/wws/arc/concat/2011-02/msg00025.html`, accessed May 1, 2011

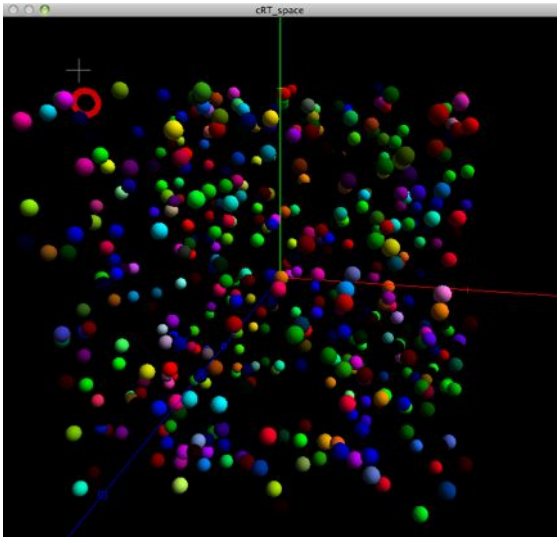


Fig. 3.4 openGL implementation for CataRT visualization

translated using mouse and keyboard shortcuts. However, the way the sound units can be directly selected by the user is still through pointing each one of units, as in the upper-left circled-in-red unit.

In terms of documentation, CataRT has extensive examples and help files, online material and documents (Schwarz 2011), and there are several publications (Schwarz et al. 2006), (Schwarz 2006), (Schwarz 2007), (Schwarz et al. 2008), (Schwarz et al. 2007) related to their principles, development, and scope. By means of these resources, the authors encourage third-party development and artistic applications using the system. They also maintain an active mailing and support list, with a large group of researchers and performers using and developing with and for CataRT.

timbreID

timbreID is a toolkit for PureData (PD) consisting on a collection of externals that allow the extraction, analysis and classification of timbral features, as well as their the storage and management (Brent 2010). timbreID provides an open analysis architecture where the user can chain objects and generate mixed feature lists according to the needs of a specific project or research. The external objects in the timbreID collection work independently. Therefore all buffering, blocking, and windowing can be performed by each single object.

Moreover, these externals work on a per-request basis, being active only when needed. Thus, the segmentation of the signal can be done by means of onset detection of the input signal, for example.

The features that the PD external objects can process range from basic spectral attributes, such as *magnitude*, *brightness*, *centroid*, *flatness*, *flux*, *irregularity*, *kurtosis*, *roll-off*, *skewness*, and *spread*, to more complex characteristics such as *cepstrum*, *mel-frequency cepstral coefficients* (MFCC) and *bark-frequency cepstral coefficients* (BFCC). These last three, cepstral-based externals are the most powerful tools for timbral classification and comparison in timbreID (Brent 2009). Each one of these external objects report the magnitude of the feature they analyze as a value or set of values.

To store, cluster and classify all feature vectors coming from the feature extraction externals, the *timbreID* external object manages and provides access to its internal database. This object has three inlets, one for each one of the following modes: *training*, *identification*, and *concatenative synthesis* (Brent 2011). In training mode, timbreID can process lists of features already extracted with the objects for feature extraction, and create a training dataset. This feature dataset of previous instances can be stored and used for future comparison and classification of new sound unit's features. In comparison mode, a sound unit's list of features is compared with the trained dataset. The timbreID external object looks for the best matching instance in its database according to given, user-defined weights for each feature in the set of features. Thus, the timbreID package is especially well suited for timbre clustering and classification. The third mode of the object also compares list of features of a given sound unit with the training dataset, as the identification mode does, but this process of identification is especially designed for concatenative sound synthesis. However, there is no documentation about its specific characteristics.

The timbreID external objects selects the closest sound unit to a target feature vector using an user-defined k-nearest neighbour approach. Four distance calculation metrics are available: *euclidean*, *manhattan*, *correlation*, and *cosine similarity*. A feature data normalization process can be selected by the user to calculate distances in a common multi-dimensional space when using more than several features (Brent 2010).

timbreID can plot the timbre space of sounds in its database according to user-chosen features for the x and y axis. The colour of the points representing the sound units can also be assigned to a feature. This 2D timbre space can be browsed and played with the mouse by pointing over each one of the sound units. The system also allows to zoom in and

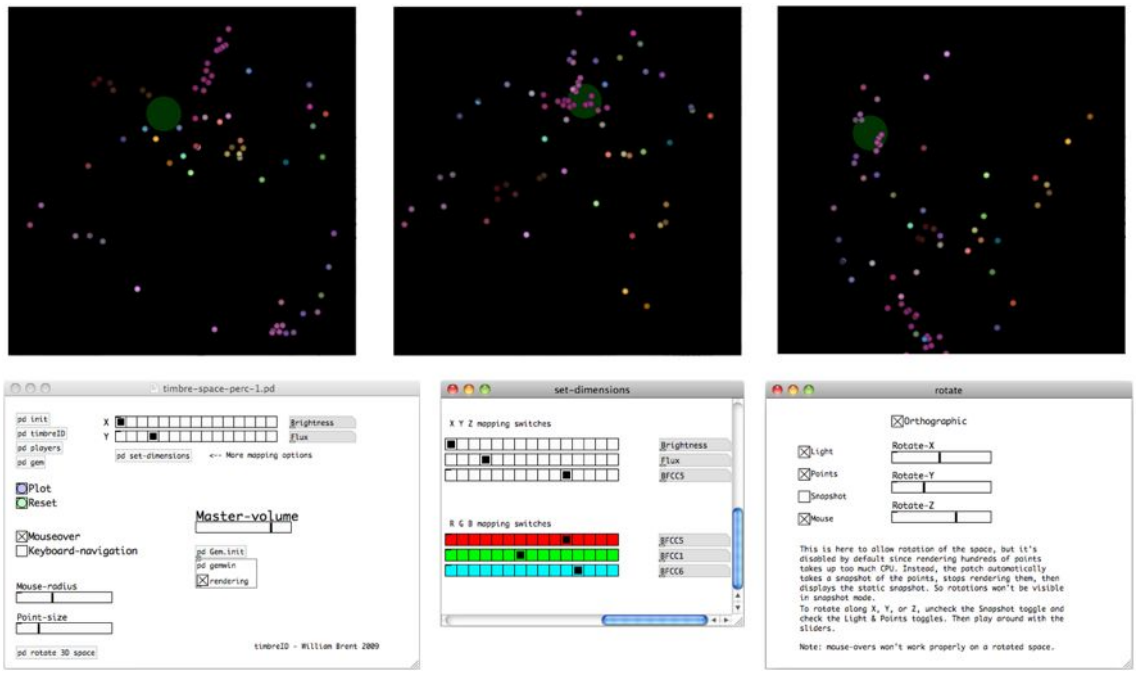


Fig. 3.5 Three rotated views of the same timbre-space using timbreID

out on a specific zone and to rotate the space in 3D, however when the space is rotated, it is not possible to browse the 3D space with the mouse. Figure 3.5 shows three rotated views of the same timbre space using the TimbreID visualization capabilities.

3.3 Interaction and Control with CSS systems

CSS systems provide access to a large number of dimensions with descriptors of different classes representing the identity of all sound units in a sound corpus. However, it is common that users of these system access and control them mainly through 2D screen browsing using a mouse to moving the target point in the descriptor space, or by using knobs or faders from MIDI controllers to control the descriptor ranges and set up more than two target descriptor values (Schwarz et al. 2008). Although these approaches provide great interaction possibilities with CSS systems and the multi-dimensional spaces they represent, 2D representations can not express the complexities of the shape and spatial distribution of the sound corpus. However, some applications have been done to perform with these systems during the last years exploring other ways of interaction with CSS systems.

Plumage (Jacquemin et al. 2007) is a 3D audio/graphic interactive interface. Its concept relies on the metaphor of controlling the position of a tape head to play "feathers" distributed in a 3D space. Each one of these plumes have an unique associated sound unit, and their position, colour, texture and rotation in the space can be related to the descriptors of the sound units they represent. Thus, the user can control the path of the tape head to active processes and play sound units once, make loops, or link many feathers to follow a score or create new sonic trajectories.

The architecture of Plumage is organized with an audio model controlled by CataRT and a graphic model controlled by the Virtual Choreographer (VirChor) software [6], an open-source, interactive, real-time engine for 3D rendering. A set of the sound units' descriptors allocated in CataRT are passed to VirChor. These values are mapped to the position, colour and rotation of the feathers in the graphical environment. VirChor, on the other hand, reports to CataRT which units have been selected by the user, and the position, velocity and acceleration of the tape head. Finally, CataRT controls all synthesis and sonic transformation of the sound units.

Plumage offers new ways of visualisation of a sound corpus as well as interesting ways of control of a CSS system. However, the interaction the system provides is done only by browsing the 3D space using a mouse and keyboard, limiting the user experience to this common approach.

The *Enlightened Hands* (Vigliensoni 2010) is a simple input device tool for gesture acquisition and a mapping strategy to explore a 2D sound descriptor space by means of open-air hand gestures. The system relies on the tracking of IR LEDs located in the tip of a performer glove's middle finger, using the IR camera of two Nintendo Wiimotes. Figure 3.6(b) shows a detail of one glove with the IR LED on the tip of the middle finger. x and z positions for each blob are extracted directly from the controller's camera data by using the OSCulator[7] application, and the y value is calculated using the triangulation method. All the synthesis part of the Enlightened Hands is performed using CataRT. The values of x and z for each hand are scaled and mapped directly into two 2D descriptor spaces using two instances of the `catart.lcd` module. The y value controls the output volume of the system. Although interesting musical results[8] can be achieved with the system due to the

[6]The Virtual Choreographer software, `http://sourceforge.net/projects/virchor/`, accessed May 30, 2011

[7]OSCulator `http://www.osculator.net/`, accessed May 11, 2011

[8]See extract of *I wanna see you*, piece composed and performed with The Enlightened Hands at

freedom of interaction with two sound corpuses at the same time using open-air gestures, the blob tracking using the wiimotes lacks of stability, making the system not very reliable for performance contexts. Figure 3.6(a) shows a performer doing one take for the piece *I wanna see you*.

(a) The glove (b) Performing

Fig. 3.6 The Enlightened Hands

Recently, a system to interact and control a CSS application with non-contact gestures was implemented for the piece *Transformation* (Jensenius and Johnson 2010), composed by the violinist Victoria Johnson. The idea behind the piece is to let the performer to trigger and spatialise different sound units by means of mapping her position to a 2D descriptor space using CataRT, and *Vector Base Amplitude Panning*[9] (VBAP). Thus, the performer can react to the sonic elements generated by the mapping of her position and improvise new material according to those sounds, generating a creative feedback loop between the environment, the piece, and herself.

The system developed for *Transformation* uses computer vision techniques to map the position of the performer to a target point in the descriptor space. A single camera hanging in the ceiling of the performance space tracks the position of the performer on stage (surprisingly, the authors refers to this tracking as "absolute position in space"), and the Musical Gestures Toolbox from the Jamoma environment [10] for Max/MSP is used to create

`http://vimeo.com/10721519`, accessed June 6, 2011

[9]The Vector Base Amplitude Panning software can be found at `http://www.acoustics.hut.fi/research/cat/vbap/`, accessed May 15, 2011

[10]`http://www.jamoma.org/`, accessed May 28, 2011

modules for the video analysis. This implementation allows for a fast mapping between the performer position coordinates on stage and a target point in the 2D descriptor space using the `catart.lcd` module, allowing the performer to select and play units in real-time according to her position.

According to the authors, the system developed for *Transformation* meet the criteria they posed as requirements for their application. However, their implementation only works in a 2D descriptor space with a single sound corpus, thus not allowing the performer to interact with more complex representations and structures.

The *Grainstick* installation (Leslie et al. 2010) allows one or two users to simulate they grab a virtual rainstick in an immersive environment. By tilting the stick, the grains move from one side of the tube to the other creating sound. The direction of the tube is related with the spatialisation of the sound grains. The tube can also be shaken or struck to modify the sound of the grains.

A DTrack Motion Tracking system and a set of two Nintendo Wiimotes are used to acquire the users' gestures in *Grainstick*. A set of reflective beacons is attached to each one of the Wiimotes, allowing the tracking system to follow their position. The motion tracker data is used to obtain the position and angular direction of the controllers and to derive their tilt and the distance between them. 3D acceleration data is measured with the embedded accelerometers of the Wiimotes and send to a computer through Bluetooth. By this means, *Grainstick* is able to track the absolute spatial position of two objects in space, and analyse other gestures for further synthesis and control of parameters.

The synthesis stage of Grainstick is build using the CataRT modules. A complex organization of the navigation space is done by creating zones, allowing users to play with different sound corpora or cluster of sounds of a given database in different parts of the performance space. To enhance the immersiveness of the experience, Grainstick uses Wave Field Synthesis to spatialise the sound grains according to the direction of the stick, and provides video projections.

3.4 Comparison

This chapter has reviewed three systems for concatenative sound synthesis. At first glance these systems work in similar way, but their implementation and the characteristics and features of their analysis, selection, and synthesis stages are different. The next paragraphs

provide a discussion on each stage and a table summarizing the main characteristics of the systems is presented.

Analysis Stage

In the analysis stage, SoundSpotter performs a high-level approach, using automated strategies for adaptive segmentation according to the musical content. By its nature, these strategies do not offer a user the control of the variables for the segmentation, and can not be tailored to the specific needs of a musical piece or project. CataRT, on the other hand, provides different segmentation approaches, giving the user more control on this process of segmenting a given sound or group of sounds into units to create the sonic palette for a piece. timbreID's segmentation works in a per-request basis, i.e. the segmentation of a sound can be triggered manually by the user, or automatically by an algorithm (analyzing if the signal's energy exceeds or is inferior to a certain threshold, for instance). Hence, different strategies can be developed and programmed in timbreID to split and analyse the sounds of the user's database.

With regard to import sounds and file management of the sound corpus, SoundSpotter needs for efficiency issues that all sound units will be allocated in a single file. This constraint makes the process of designing a sound corpus with this CSS system time-consuming and not very interactive. timbreID does not provide batch processing capabilities for importing sound units, but its open architecture allows the user to implement this feature.

The three systems offer different descriptors types to express the identity if the sound units. SoundSpotter provides 89 LFCC coefficients to describe the morphological aspects of the sound units. However, this large amount of values not necessarily convey more musical information about the timbral similarity of the sound units because there are inherent correlations in these quantities. CataRT works with a smaller set of 25 descriptors, but only less than half of them refer to acoustic and timbral characteristics of the sound units. However, part of the rest of the descriptors can express human-annotated categorical descriptors of the units that can be used to create sub-sets of sounds in the corpus. Thus, CataRT lacks of richness when it tries to describe the sound units in terms of their spectral characteristics, not offering many possibilities for different timbre-space representations, but it is flexible in terms of offers other useful descriptors. timbreID provides a large set of spectral, timbral-related descriptors but it lacks of another kind of descriptors to organize

sound corpuses according to arbitrary decisions.

The selection of the best-matching sound units in the three systems is performed real-time in an unit-by-unit basis. This process not considers the concatenation quality of the sound units (Schwarz 2006). In order to narrow possibilities of finding a matching sound unit according to a target descriptor vector, CataRT and timbreID offer the user the possibility of weighting some of the descriptors. By this means, the system provides certain degree of control to the user in the unit selection process. Soundspotter, on the other hand, does not offer this feature, but it has "memory" methods that allow the system to forget already played units for an amount of time, obliging the system to search for and play new sound units. This characteristic is convenient because it allows to explore zones of the sound corpus that sometimes are not played.

In the synthesis stage, CataRT is the only one of the three systems that offers transformation methods to process the sound units. With these methods, CataRT tries to overcome the lack of concatenation quality described, creating smoother transitions between the concatenated units. Table 3.1 shows a summary of some of the most relevant characteristics of SoundSpotter, CataRT, and timbreID.

According to the previous review of features and characteristics of the three CSS systems, SoundSpotter, CataRT, and timbreID, we chose CataRT as the system to augment its interaction and control possibilities using non-contact gestures. Chapter 4 will present an empirical comparison of the performance parameters of four position tracking devices. The results of this comparison will inform the implementation of a system for touchless gestural control of CSS in Chapter 5.

Category	Subcategory	SoundSpotter	CataRT	timbreID
System	Code	Max/MSP PureData (C++ externals)	Max/MSP (FTM, Gabor, MnM libraries)	PureDaTa (C externals)
	Uses	Data-driven CSS	Data-driven CSS, User-driven CSS	Data-driven CSS, User-driven CSS, Timbre-classification
	Visualisation	No	User-defined 2D + colour descriptor space	User-defined 2D, 3D + colour descriptor space
	Documentation	Scarce	Extensive	Scarce
Analysis	SoundCorpus	Single file	Multiple files (batch process)	Multiple files
	Segmentation	Automatic variable windowing	Imported from external files, arbitrary grain segmentation, split by silence	Several windows types
	Descriptors	Morphological descriptors (LFCC, 89 coefficients)	MPEG-7-based descriptors. Unit, categorical, and symbolic descriptors. Statistical methods for time-evolving values	Morphological and signal descriptors
Selection	Distance	Euclidean distance	Mahalanobis distance	Several distance metrics
	Closeness	Closest unit	Closest or random unit in a user-defined radius	closest unit
	Memory	Long- and short-term memory loss		
	Weights	Timbral or pitch features user-weighted	Weighted features	Weighted features
Synthesis	x-Fade		User-choosable fade in-out values	
	Transformation		Several simple transformation methods	
	Catenation modes	Single-mode	User-choosable modes	Single-mode

Table 3.1 SoundSpotter, CataRT, and timbreID CSS systems summary

Chapter 4

Position Trackers, Experimental Comparison

As reviewed in Chapter 2, different techniques can be used for spatial tracking and many of them have been tested and used in music performance and composition. These techniques rely on different sensing methods and share different performance parameters and characteristics across systems. In the study of the development of a touch-less interface for musical expression, the selection of a tracking system will entail using all the characteristics, strengths, and limitations of that technology. These factors can have consequences in the performance and playability of the instrument. Additionally, to develop expert performance of playing an instrument a deterministic behaviour is decisive, so to develop a reliable interface we need to know in advance how the system will respond to the same stimuli in diverse contexts.

This chapter reviews the characteristics and intrinsic constraints that four tracking devices of different technologies have, and presents an experimental comparison of some of their performance parameters. The stationary accuracy, precision, update rate, and shape of the space these position tracking systems sense, is measured and contrasted. These results will give some clues to make design decisions for the development of a touchless gestural interface to control a CSS system in Chapter 5.

4.1 Position Tracking Systems

The compared trackers were the Vicon 460 motion capture system[1], the Polhemus Liberty 240/8 magnetic-based motion tracking system[2], the Microsoft Kinect computer-vision based system[3], and the In2Games Gametrak, a mechanical, tether-based, position tracker system (Myers 2002). While the former two devices are considered professional position trackers, the latter are consumer-oriented systems used in game consoles.

The experimental comparison of these systems was done in the same space, a music technology laboratory hosted in an office building with no special construction considerations. This issue had an effect on some of our measurements because we did not have control about the construction materials in the walls, floor, ceiling, or the dimensions of the test space. While the materials present in the room influenced only the measurement of the Polhemus magnetic tracker, the dimensions and size of the room shaped all measurements. The next subsections provide a brief summary of the technical specifications of the systems we compared.

4.1.1 Vicon 460

The Vicon motion capture system tested comprised the following components: six Vicon M2 cameras, the Vicon 460 Datastation hardware module, and the Vicon iQ2.5 and Eclipse workstation software packages.

The Vicon 460 Datastation handles all the camera synchronization, co-ordinates the capture and generation of video data, and sends the information to the Vicon Workstation and iQ2.5 softwares. Although the Eclipse workstation software allows to record, organize, analyse and present the data, we did not use it (except for storing the calibration set up) because the experiment considered to extract the position data from the system in real-time. The iQ2.5 software is capable of sending real-time data to a third-party application by means of the Targus and Real-Time Engine internal applications. The M2 cameras can operate at several frame rates, ranging from 60Hz to 1000Hz, but for the experiment an update rate of 250Hz was chosen because most of the tests are related to stationary measurement and high rates are not required. With this setting the active resolution of

[1]Vicon Motion Capture Systems, `http://www.vicon.com/`, accessed May 1, 2011

[2]Polhemus Liberty 8 Electromagnetic Motion Tracking System, `http://www.polhemus.com/?page=MotionLiberty`, accessed May 1, 2011

[3]Microsoft Kinect for Xbox 360, `http://www.xbox.com/en-CA/kinect/`, accessed May 1, 2011

the camera is set up at 828 * 656 pixels (H*V) with an aspect ratio of 1:0.79. Table 4.1 shows some of the technical specifications of the system we used running at 250Hz.

Property	Specification
Number of supported video camera channels	6
Update Rate	120Hz
Field of view	42°H, 34°V
Image size	828 * 656 px
Spectral Infrared Sensitivity	100% @875nm
Host Workstation OS	Windows
Operation Environment	Indoor

Table 4.1 Vicon 460 MoCap System with M2 cameras @250Hz technical specifications (Vicon Motion Systems 2002)

4.1.2 Polhemus Liberty 240/8

The Polhemus Liberty 240/8 is an electromagnetic-based tracker system. It features an electronic control unit with the hardware and software to generate and sense magnetic fields, it measures position and orientation, and communicates with a computer. The tested system has 8 sensors with 6DOF and makes 240 measurement updates per second, per sensor, according to its specification sheet. For our experiment, we used the Polhemus Long Ranger source, which allowed us to have a larger sensing area of about three times (see Table 4.2).

4.1.3 Microsoft Kinect

There is no official information from Microsoft about the technology behind the Kinect. However, PrimeSense, the company that provides the raw tracking technology to Microsoft, has released reference design information about the PrimeSensor^{TM} (PrimeSense 2011), the Kinect base technology, and the patent of the system is also available (Freedman et al. 2010). This references provide information about how the Kinect system works.

The Kinect system comprises the camera and software that processes all data acquired by the camera. The system does two things: it generates a 3D image of all objects in its field of view, and it also recognizes human beings among those objects (Carmody 2011).

Property	Specification
Degrees-of-freedom	6
Number of sensors	8
Update Rate	240Hz (per sensor)
Static Accuracy Position	0.08cm
Static Accuracy Orientation	0.15°
Latency	4ms
Max. range (standard source)	1.52m
Max. range (Long Ranger source)	4.6m
Interface	RS-232 / USB
Host OS	Win2000/XP
Operation Environment	Indoor

Table 4.2 Polhemus Liberty 240/8 technical specifications (Polhemus Liberty webpage, accessed May 11, 2011)

The 3D image is created by illuminating a scene with patterns of in infrared light. The reflection of these patterns change depending on the distance of the object were the light is reflected and the device. The reflected light is acquired by the system and the patterns are analysed. With this information, the Kinect is capable of reconstructing a depth map of the image. In this moving image, the system looks for shapes that can resemble the human body and, when it found them, looks for specific body parts or joints. When they are detected, the Kinect begin to track them and calculate their position (Joystiq 2011).

The PrimeSensor has an IR light emitter, a VGA depth image camera with CMOS sensor chip, an UXGA colour image camera with another CMOS, and a set of two microphones. Figure 4.1 shows a block diagram of the PrimeSensor^{TM} and their constituent parts. Its technical specifications are shown in Table 4.3.

4.1.4 In2Games Gametrak

The Gametrak^{TM} "direct motion capture system" is a two-joystick, tether-based system capable of measuring 3D position of up to two points in space. The two joysticks of the system are based on analog potentiometers that measure the x and y position for each one of the two points in space. To calculate the z position, another two potentiometers are used to quantify the distance of the extension of nylon tethers that pass through the tip of

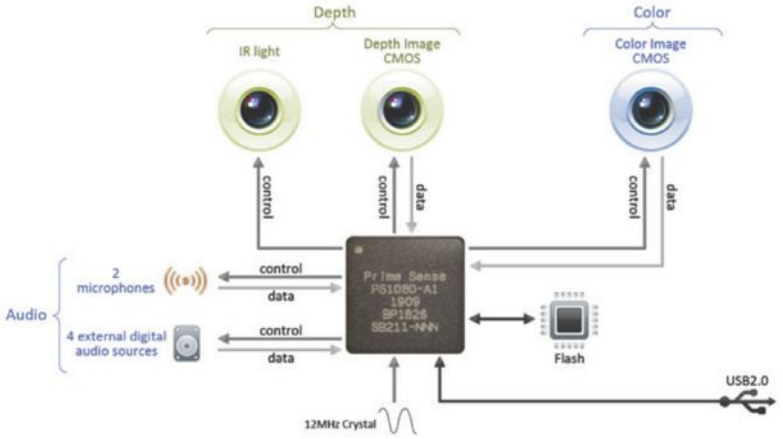


Fig. 4.1 PrimeSensor^{TM} block diagram (Primesense 2011)

Property	Specification
Field of view	58°H, 45°V, 70°D
Colour image size	1600 * 1200 (UXGA)
Depth image size	640 * 480 (VGA)
Spatial x/y resolution (@2m from sensor)	3mm
Depth z resolution (@2m from sensor)	1cm
Maximum image frame rate	60fps
Operation Range	0.8m - 3.5m
Data Interface	USB 2.0
Operation Environment	Indoor (every lighting condition)

Table 4.3 PrimeSensor^{TM} technical specifications (Primesense 2011)

the joysticks by counting the number of turns each potentiometer does. The nylon tethers are rolled in a retractable spring-loaded drum (Myers 2002). The implementation of the system seems very simple, but a complex mechanical system to guide the nylon tethers in order to have a clean path is used (Freed et al. 2009). The housing of all potentiometers, nylon tethers, interface, and joysticks is a weighted box that allows the user to pull out the chords without moving the box.

The Gametrak system converts the acquired spatial data using the USB-HID (Universal Serial Bus - Human Interface Device) protocol for three systems: XBOX, Sony PS2, and personal computers.

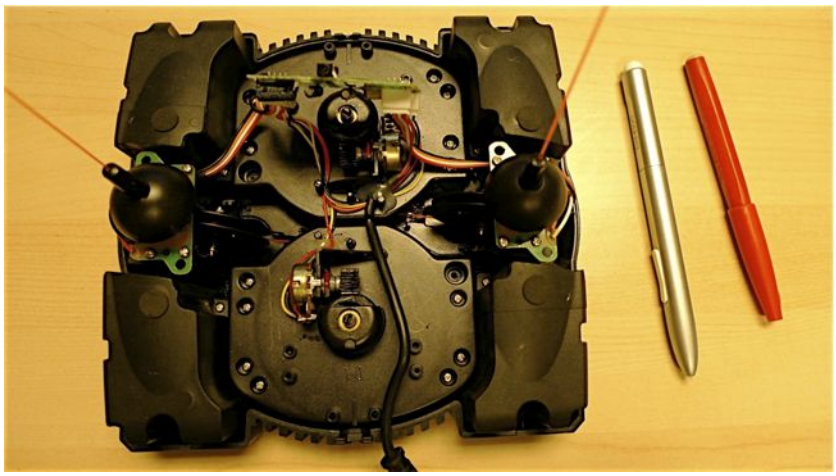


Fig. 4.2 In2Games Gametrak

Property	Specification
Degrees-of-Freedom	3
Number of Sensors	2 + Footswitch
Resolution	12 bits
Data Interface	USB-HID
Host	XBOX / PS2 / PC
Operation Environment	Indoor

Table 4.4 Gametrak technical specifications

4.2 Experiment design and set up

To compare the reported data by the mentioned trackers, we used the experimental approach described by Kindratenko (Kindratenko 2001) to compare the accuracy of two tracking systems. He collected the reported data by each tracker at known, nominal, two-dimensional locations, and calculated their accuracy in those points. Although his results are well documented and clear, his experiment only dealt with measurements taken on a plane. Hagedorn et al. (Hagedorn et al. 2007) designed another experiment to correct the data measured by an electromagnetic motion tracking system. They developed a three-dimensional grid by means of crates, and collected the position tracker reported data at known locations. Because of the nature of the magnetic-tracking system, they use plastic crates so as not to distort the measurements of the tracker.

For our experimental design, we used a combination of the aforementioned approaches to plan our experiment. Although the location of points for the measurements could be anywhere in the space, it is easier to have all points in a known grid. Hence, we created an evenly spaced grid on the floor and used lockable, plastic crates to have the same grid at different heights. The dimensions of the crates were 31.5 cm * 42.7 cm * 26.4 cm, and we used five crates for each side on the floor, x and y, and four crates for the z axis, creating a volume of 1.78 m * 2.14 m * 1.06 m. The two-dimensional grid on the floor and crates on the grid are shown in Figure 4.3.

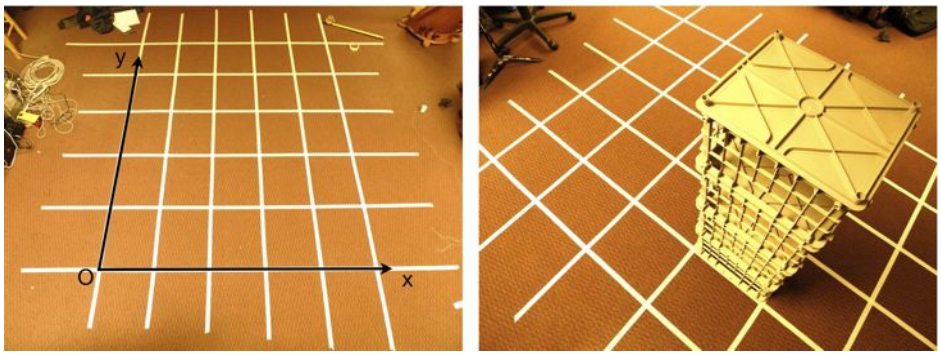


Fig. 4.3 2D grid and crates for 3D measurements

We located the trackers at the best possible position to see all points in the grid. However, the dimensions of the room and the system's minimum and maximum tracking distance did not allow us to place all trackers far away enough from the grid, so some points

on the grid were not seen by some trackers. As we needed to compare all four systems in the same space, we opted to reduce the sensed volume and measure only the rectangles located at the centre of the grid, resulting in a space of dimensions (0.95m * 1.28m * 1.06m), and volume of 1.29m^3. To cover this space, we did 80 measurements at the corner of each plastic crate. The measurement was not done by level, but for column in a specific order, as shown in Figure 4.4 (column three is not shown).[4]

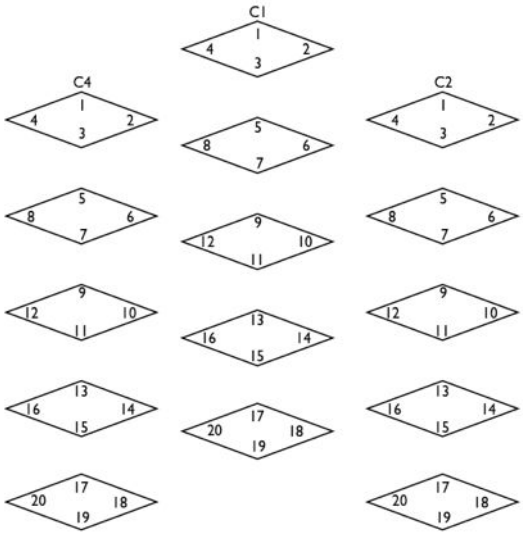
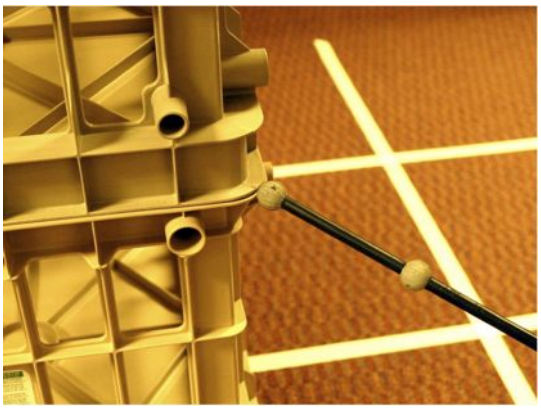


Fig. 4.4 Measurement sequence

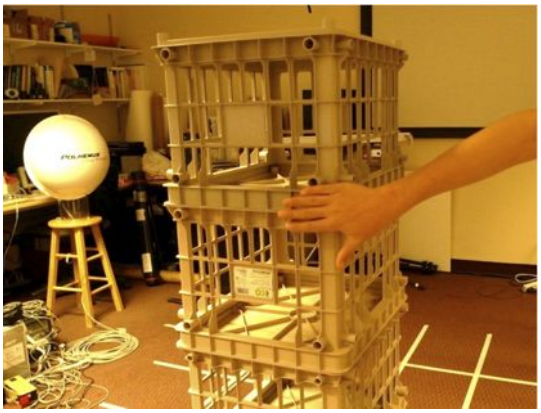
For the measurement of each of the 80 points in the 3D grid, the different markers or sensors for each system were places on each vertex of the plastic crates. As can be seen in Figure 4.5, for the Vicon system we used one of its IR-reflective beacons, for the Kinect it was a human hand, for the Polhemus we used one of its sensors, and for the Gametrak we used the tip of one of the system's nylon tethers.

In order to measure the accuracy, precision and update rate of the systems, we recorded the tracker's reported data during the lapse of one second. We then took the mean of the reported values and obtained a more representative point to find interpolated lines along each axis and plane. Considering all values in one second also allowed us to calculate the precision of the tracker for those given points. Also, subtracting the arrival time of

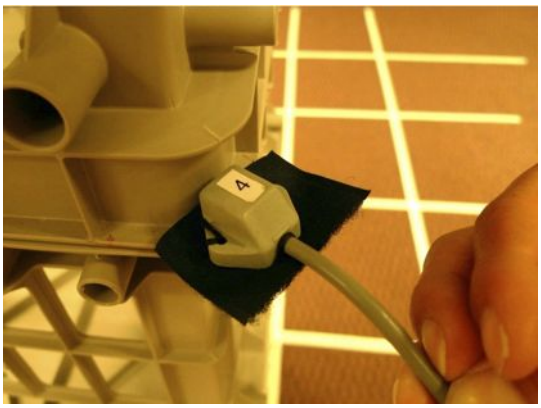
[4]It should be considered that several errors can affect the position data acquired by the trackers. These errors can range from a non-uniform crate manufacture, to human-error in the creation of the grid, in the placement of the crates on the grid, and in the positioning the sensors at the measured points in the crates.



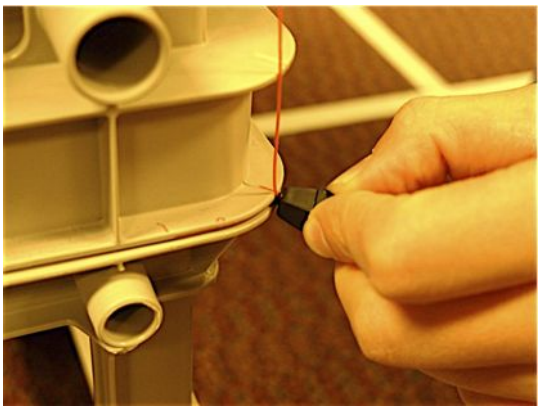
(a) Vicon



(b) Kinect



(c) Polhemus



(d) Gametrak

Fig. 4.5 Measurement of the same point in space with the four tracking systems

consecutive measurements gave us the value of the update rate, and we saw how stable it was over time.

As we located the trackers in different parts of the room to find a good position to measure the space, their origin was located at different points. Thus, a translation and rotation of the vectors they reported was performed to position all trackers at the same virtual place, providing them with the same point of view. To achieve our goal, we measured the vectors going from the tracker-origin to the measurement-space-origin and axes of the plastic crate located at the centre of grid. We then calculated the new unit vectors $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ using a matrix *change of basis* to translate and rotate the spaces from the tracker space to the new calibrated space (see section 4.2.5 for more details).

Since the systems reported their data in different ways and different scales, for doing a proper comparison with similar characteristics, a workflow pipeline was designed. It is worth mentioning that the results of the experimental comparison will be related to the whole workflow pipeline that includes a tracker and its data flow and processing. The pipeline is shown in Figure 4.6.

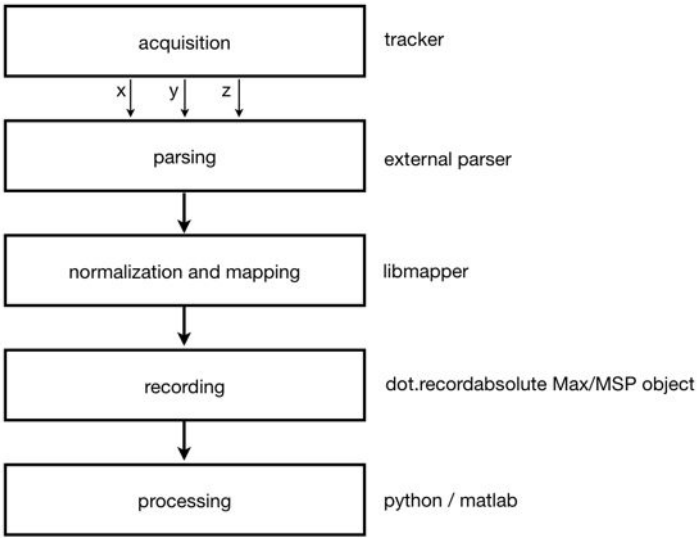


Fig. 4.6 Position trackers workflow pipeline

4.2.1 Data Acquisition

The data acquisition stage refers to the process of setting up, calibrating, and recording data from each one of the systems. Because each of the position trackers has its own sensing method and characteristics, different approaches were taken for each one of them.

Vicon

The Vicon system has the most detailed and complex calibration method of the trackers we tested. The cameras of the system have to be properly positioned to cover all the sensed space, and all major light reflections (for instance, from shiny metal objects) ought to be removed from the camera view before starting the process of calibration, otherwise the calibration procedure fails. There are two processes of calibration, dynamic and static. The dynamic calibration involves moving a wand with three reflective markers all over the

sensed space with roughly "figure-of-eight" movements. This procedure ensures that the markers can be seen by at least three cameras in all the experimental space. Thus, this process allows the system to determine the sensed space. The static calibration requires the use of a calibrated rule on the floor for setting up the axes and unit scales of the system. Once done, the system is capable of telling the user if the calibration was successful and can be stored for future use. In the acquisition stage, the information from the 6 cameras is processed by the Vicon data station, which is sent to the iQ 2.5 software. An internal application called Targus connects the real-time engine of the software for sending 3D position and orientation data for all defined objects with a minimum set of three markers.

Kinect

Since its release in 2010, different groups of people have been working on ways to extract the tracking data that the Kinect can track from a user's body. We informally tested several systems and ended up using the OpenNI framework library [5] because it provides the shortest latency between the acquired position and the data the system reports. The Kinect system, in companion with the OpenNI framework library, has a calibration procedure that needs a specific user's pose. After this process, the system is capable to track 15 different points located at the end of the user's limbs, or in between those points, like for tracking the neck or knees. Figure 4.7 shows the user's pose needed for the calibration pose of the Kinect and the fifteen reported points of a user's body.

The Kinect performs a proprietary signal processing to determine where these points are located, and it was not possible for us to know where the system was exactly measuring during the calibration process or the data acquisition stage. Figure 4.5(b) shows how the Kinect system measured all 3D grid points: by placing the centre of the tracked hand around the position of the measurement. It is not clear, however, to know what point the system is exactly looking at. We tried to trick the system to have a static, more reliable, measurement by means of using a non-human structure making the required pose, but the times that we fooled the system was minimal in comparison with the failed calibrations.

[5] OpenNI Framework, `http://www.openni.org/`, accessed May 1, 2011

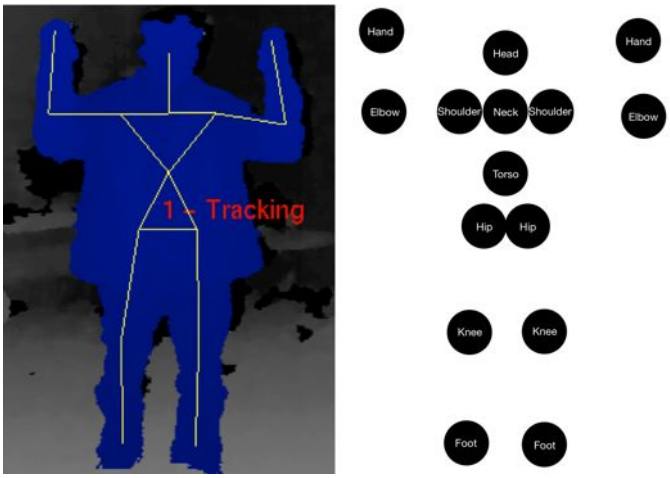


Fig. 4.7 Microsoft Kinect calibration user's pose and tracked points

Polhemus

The third compared system, the Polhemus magnetic tracker, ideally requires a "benign" environment, free of ferromagnetic materials or metallic surfaces. Otherwise, the measurements will suffer from distortion. As we did not have this ideal space we tried to reduce the amount of metal structures and objects in the room. However, we were not sure about how much metal exist in the floor, ceiling, and walls of our laboratory. Moreover, as we wanted to test these systems for performance situations, extracting and sending their reported measurements in real-time, we could not use *PiMgr*, the Polhemus software, which can reportedly compensate for the distortion from metal objects according to a compensation map, since it does not have a native data packager to OSC.

Gametrak

The Gametrak mechanical tracker does not have any calibration setup and it is mostly ready for instant access to the data reported by its two "three-dimensional" joysticks. However, the version of the device we got was developed for being used in Sony PS2 consoles, so, we followed a Gametrak hack by Mesker[6] and modified the device to extract its reported joystick data. Figure 4.8 shows the circuit and the two bridged points to be capable of retrieving the data on personal computers.

[6]Mad Catz Gametrak Mod, textttthttp://x37v.com/x37v/post/labels/sensors.html, accessed May 1, 2011

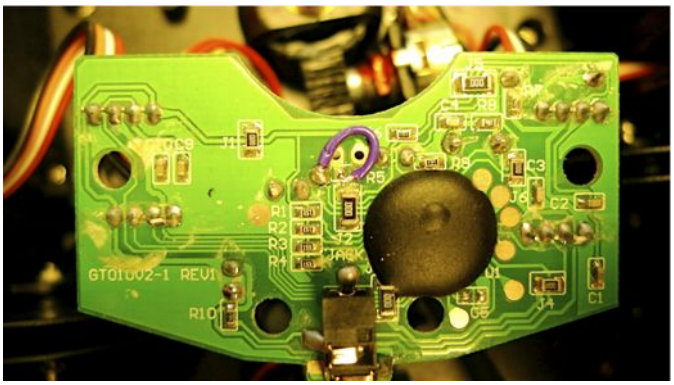


Fig. 4.8 Gametrak modification

The Gametrak requires the user to grab their plastic tethers. However, the device itself is light, so for a proper measuring it must be solidly attached to a surface to avoid changes in its position. In our test, we fastened the tracker to the ceiling of the room by means of a set of bar clamps. This set up also helped us to solve the problem of measuring points behind any of the boxes, which were out of the direct line-of-sight of the device.

4.2.2 Data Parsing

To record and handle all tracker data we used a different computer from the one for the data acquisition. Therefore, we needed a way to extract, parse and send all data from the trackers to the recoding computer in real-time as OSC messages.

Vicon

To extract the data from the Vicon system, we followed previous research (Eckel et al. 2009) and used the *QVicon2OSC* application[7]. This application bridges the Vicon motion capture data to OSC, and is capable of sending the position of user-defined points and the rotation of objects created by the user. For the purposes of our experiment we were interested only in the position of a point. Once structured and sent as an OSC message, the data was received in Max/MSP with a customized version of the *OSCeletontoQC* object[8].

[7] *QVicon2OSC*, Vicon motion capture to Open Sound Control bridge application. `http://www.sonenvir.at/downloads/qvicon2osc/`, accessed May 1, 2011

[8] *OSCeletontoQC* Max/MSP object available for download at `http://mansteri.com/download/software/osceletontoqc/`, accessed June 7, 2011

Kinect

To retrieve the data from the Kinect system, we used the *OSCeleton* software[9], an OSC proxy for Kinect Skeleton data. This software establishes communication with the OpenNI framework, which communicates with the device. OSCeleton allows a user to scale and offset the data, but we opted for not using these features because we would perform a data post-processing in a later stage of our workflow pipeline. The OSC data was also received in Max/MSP.

Polhemus

To acquire and parse the data from the Polhemus Liberty tracker to OSC messages, we used the library and command-line front-end *plhm*[10]. This application is capable to request data from the Polhemus Liberty and send it through a network as OSC messages. plhm does not allow the user to perform any kind of compensation or scaling of the data. Once parsed as OSC messages, we received the Polhemus data in Max/MSP.

Gametrak

Once the Gametrak device was hacked, its acquired data could be opened directly in Max/MSP because the device is natively recognized as an HID object. The reported raw data was formatted as an OSC message for further processing.

4.2.3 Data Normalization and Mapping

As the data reported by each of the trackers has its own scale, we normalized it so that it is possible to compare them. At the same time, as we were handling 3D position data from several trackers, a tool for flexible mapping was required.

We used the *libmapper* library and its *MapperGUI* Max/MSP external[11] to map and scale all signals from the trackers to the recording computer. This library is capable of discovering devices in a network and showing their previously declared inputs and outputs.

[9] *OSCeleton*. `https://github.com/Sensebloom/OSCeleton`, accessed May 1, 2011

[10] *plhm*, a library and command-line front-end for acquiring data from Polhemus motion tracking devices. `http://idmil.org/software/plhm`, accessed May 1, 2011

[11] *libmapper*, a library for representing input and output signals on a network, allowing arbitrary mappings. `http://idmil.org/software/libmapper`, accessed May 1, 2011

A user can arbitrary map those input and output ports with any kind of scaling of the signals going through the mapper. In our experiment, we measured the reported maximum and minimum data by each one of the trackers and used the normalization capabilities of the libmapper to send values between -1.0 and 1.0. The normalized output of all trackers was routed again to Max/MSP.

4.2.4 Data Recording

To record the normalized data from all trackers we used the *Digital Orchestra Toolbox's* `dot.recordabsolute` Max/MSP object (Malloch et al. 2011). This object allows a user to record arbitrary number of data streams, such as OSC messages, with absolute time-stamping. This capability to record the timing of the messages was considered of the utmost importance for the experiment because one of our goals was to compare the update rates of the systems.

4.2.5 Data Processing

In terms of data processing we performed two steps. The first step was to parse all data to a file format with the proper internal structure to be processed. The `dot.recordabsolute` object saves data in text files starting from the last recorded moment. However, to have all files from past to present, ordered by columns, and convert them to comma-separated values (CSV) files, we developed a Python script to parse and arrange the data.

After the creation of the CSV files, we performed a translation and rotation of the spaces in a process named *change of basis*. This process allowed us to virtually locate each tracker's origin at the same point by converting the vectors measured in one basis (the tracker basis) to another one (the normalized-space basis). We could then measure the vectors for each point in the space with the same, normalized reference space. The change of basis procedure we implemented is as follows:

Supposing that we have two different bases

$$\mathcal{B} = (\vec{v_1}, ..., \vec{v_n})$$

and

$$\mathcal{C} = (\vec{w_1}, ..., \vec{w_n})$$

we can express the same vector (equal magnitude and direction) having different representations relative to the bases $\mathcal{B}$ and $\mathcal{C}$, as

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}_{\mathcal{B}} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}_{\mathcal{C}}$$

Thus, defining D as a matrix whose columns are the vectors from basis $\mathcal{B}$

$$D = (\vec{v_1} \cdots \vec{v_n})$$

and G as a matrix which whose columns are the vectors from basis $\mathcal{C}$

$$G = (\vec{w_1} \cdots \vec{w_n})$$

Then

$$G^{-1}D \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}_{\mathcal{B}} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}_{\mathcal{C}} \; and \; D^{-1}G \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}_{\mathcal{C}} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}_{\mathcal{B}}$$

Which means that $(G)^{-1}D$ converts from basis $\mathcal{B}$ to basis $\mathcal{C}$, and $D^{-1}G$ converts from basis $\mathcal{C}$ to basis $\mathcal{B}$.

We implemented a function to perform the translation and change of basis. To calculate the translation, we subtracted the mean of all measurements over one second between the vector going from the tracker-origin to the new origin, and the mean of all vectors going from the tracker-origin and each one of the unit vectors in the normalized-space. We then calculated a new origin with $\hat{\imath}$, $\hat{\jmath}$, and $\hat{k}$ as the unit vectors in the calibrated space. In this new reference space, an interpolated function between the mean of all measurement for each one of the points in the spatial grid was calculated to visualise the sensed space, and box and whiskers plots were generated to visualise the accuracy, precision and update rate of each one of the tracker systems.

4.3 Results

In the following pages, we present a summary of the experimental results. They are grouped by shape of the reported space, accuracy and precision, and tracker update rate for each one of the four systems.

4.3.1 Reported space

We measured points at discrete places in the space, over a lapse of one second. The amount of reported values depended on the update rate of each one of the trackers. To visualise the shape of the space reported by the trackers, we interpolated lines between points on each axis per plane of measurements.

In the following pages we present the results for the reported space by all trackers. Red dots in the plots represent the measurement for each point. The closer the points are to a nominal, integer value, the more accurate the system is; and the smaller of the red zones are, the larger the precision is. Furthermore, the straighter the blue lines are, the less distorted the space representation is. The top plot represent a 3D, isometric view of the reported space. The three plots at the bottom represent XY, XZ, and YZ views. The scale of the plots is normalized, (the value for each axis is equivalent to each side of our calibration object, i.e. a plastic crate (31.5 cm * 42.7 cm * 52.8 cm)).

Vicon

Figure 4.9 shows the measured space by the Vicon. It can be seen that the reported space by this tracker is very close to the original. We measured points in a 3D rectangular grid and what the system reports is what we measured. The blue lines are mostly straight, making clear that the Vicon system senses the space evenly along its axes. However, we can see in the YZ plot of the figure that the points located between $z = [2 : 4]$ at $y = [1 : 2]$ are slightly shifted to the right. Analyzing the XY plot, we can see the same shifting in those points, but moving along with the x values. After measuring the floor with a level, we realized that the ground was slanted in that region of the room, altering slightly the vertical position of the column of crates and generating a shift in the x and y values.

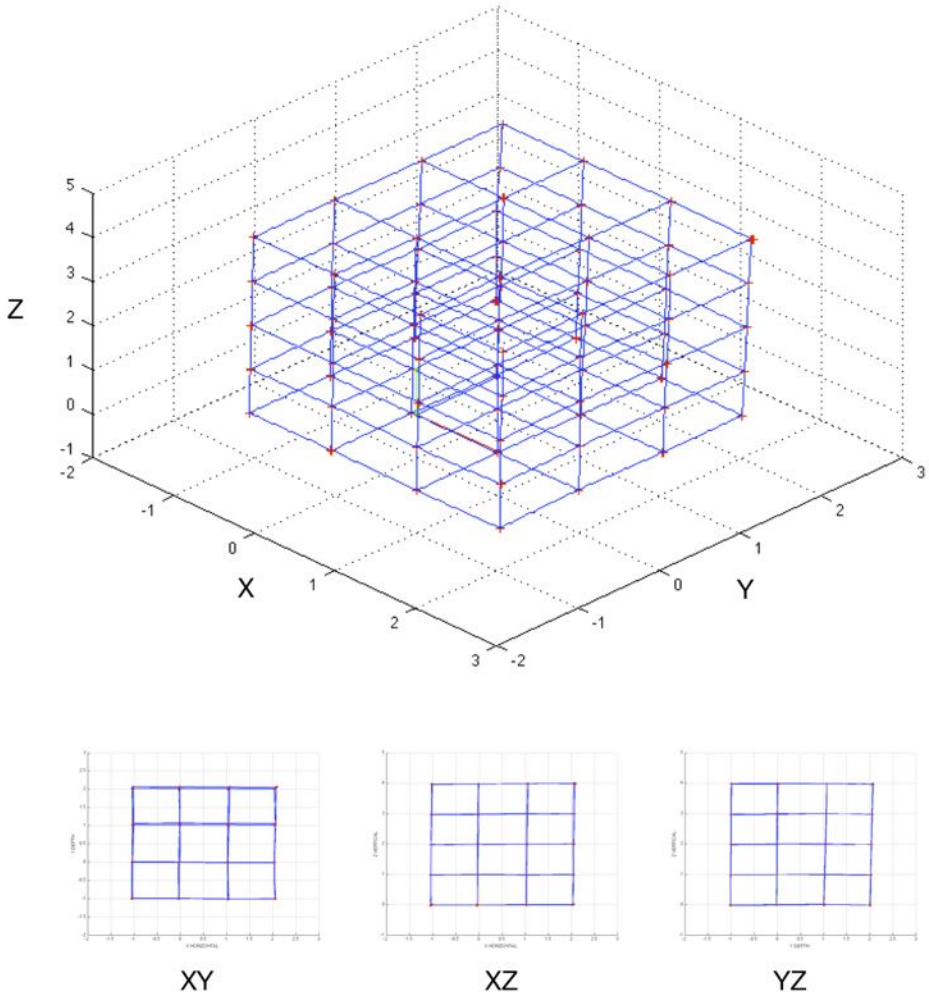


Fig. 4.9 Vicon 460 reported space

Kinect

Figure 4.10 shows the reported space by the Kinect. We can see that along the three axis, the measured points are not located in the nominal positions, making the reported space very curved, and showing that the tracker is not accurate. Also, there is a large variability in the position of the red points, so this system is far less precise than what we see with the Vicon. However, the overall shape of the sensed space is still a deformed, wavy cube.

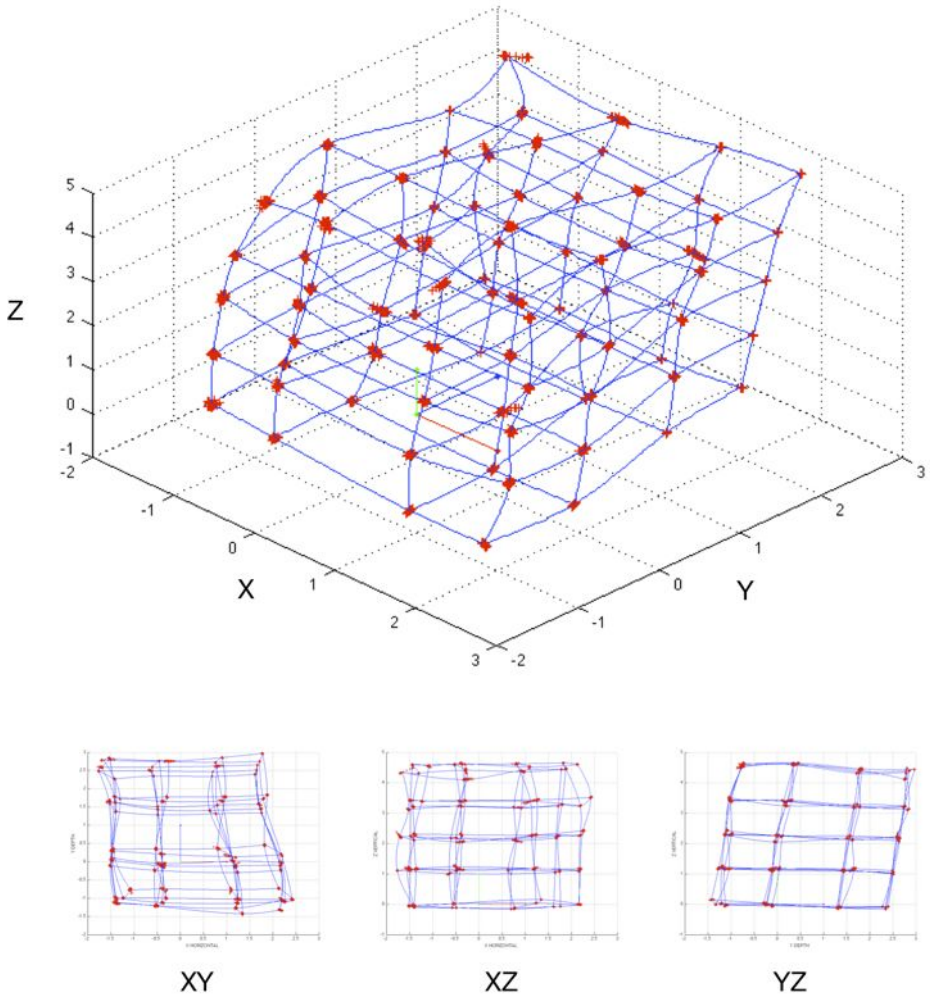


Fig. 4.10 Microsoft Kinect reported space

Polhemus

Knowing in advance the distortion that ferromagnetic and metallic surfaces create in the measurements of magnetic-based position trackers, before testing the Polhemus we removed as much as possible all metallic devices and elements in the room and placed the crate for the calibration stage in the middle of the height of the room.

Figure 4.11 shows interpolation lines between the reported points by the Polhemus magnetic-based position tracker. The shape of the space measured by the Polhemus in our test-room, with its unique conditions, was very particular. It can be seen that the middle-line along the x and z axis are relatively straight. Also, lines along y, close to the magnetic source, start straight but bend at the end. However, the most distant-to-the-source points were reported far away from their nominal position. Equally, the red points closer to the magnetic source shows less variability than those located at a larger distance. This issue can be explained by the presence of ferromagnetic elements in the floor and ceiling of the room that we could not remove. Also, the magnetic field created by the source decreases with the distance, so the special conditions of the room distort more profoundly the measurements at those points, shifting their reported position. The dimensions of the space in which the measurements were accurate to a certain degree are (94.5 cm * 85 cm * 79.2 cm), equivalent to a volume of 0.64m^3.

Although we were expecting a bigger linear space, our results were good in the sense that they represent a common room, with unknown metallic surfaces and conditions. This kind of rooms are a frequent environment for music performance, which is the context of our research.

Gametrak

Figure 4.12 shows plots for the reported space by the Gametrak. We can see that the red points were not located in the nominal position of the measurement, but in a "radial" cube. This rectangular shape was scaled-dependant on the distance from the tracker to the level which was measured, and its value became larger for longer distances. This unexpected behaviour was rather consistent for all points in the measured space. Also, the red points were more sparse than in the Vicon or Polhemus, indicating that the system is less precise than those trackers, as expected.

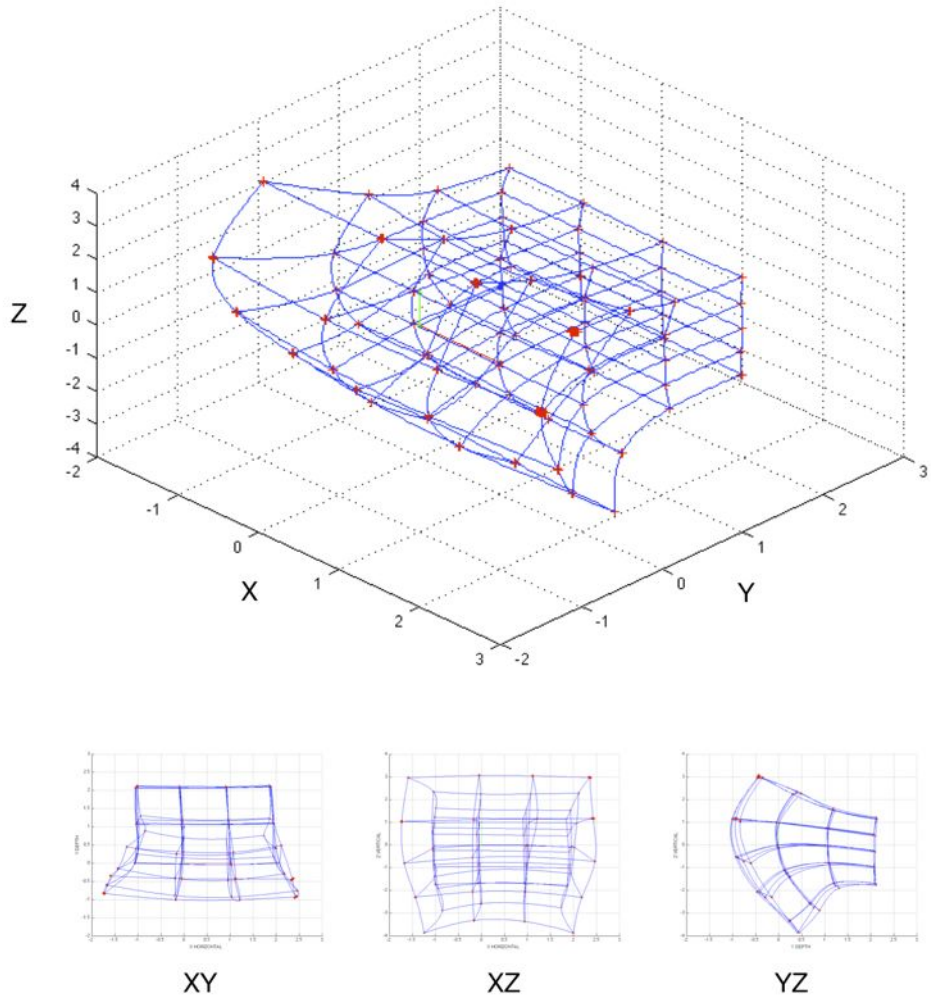


Fig. 4.11 Polhemus 240/8 reported space

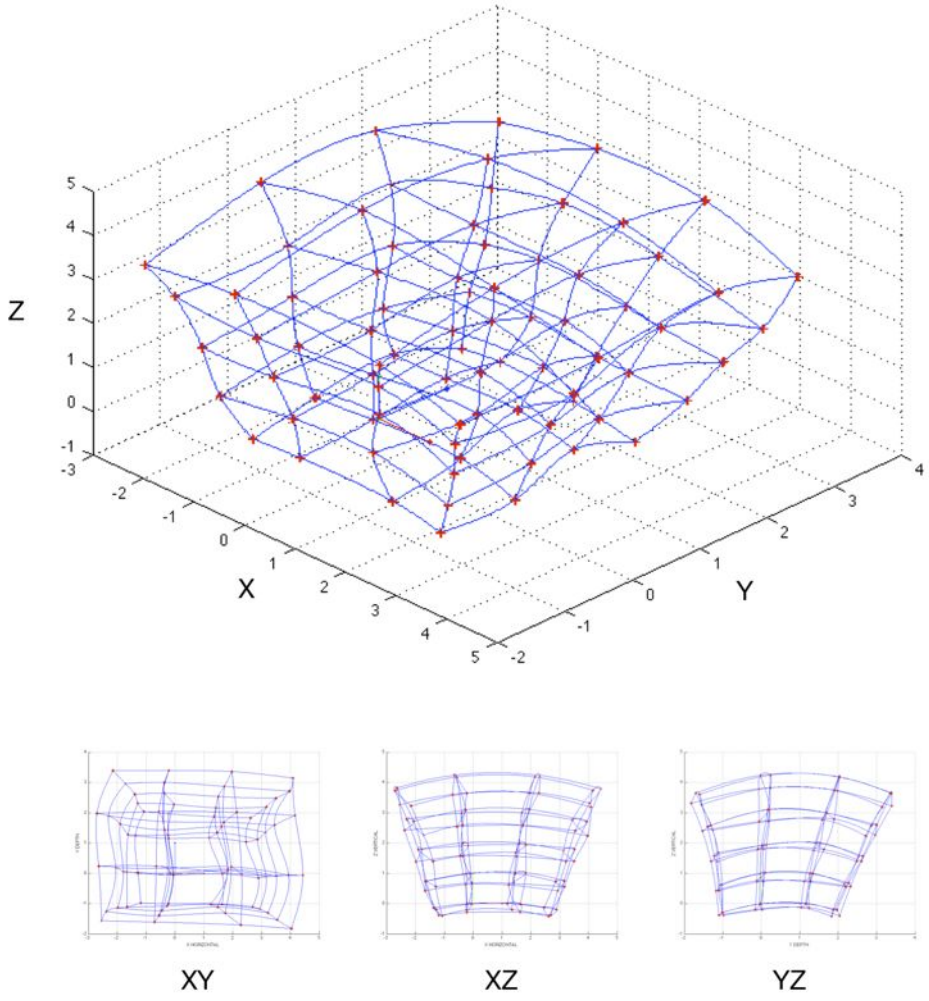


Fig. 4.12 In2Games Gametrak reported space

4.3.2 Accuracy and precision

In our experiment we did the measurements at discrete points inside a normalized space. To evaluate the accuracy of each system, we calculated the mean of all measurements, and compared it with the nominal, actual value of the position we measured. The closer the mean value to the actual integer point meant a higher accuracy of the system. At the same time, small deviations of the measurements compared with the nominal position, implied a higher precision.

Box-and-whisker plots show the median of the data (central line), the lower (Q1) and upper (Q3) quartile (lower and upper lines of the box), the smallest and largest observation (lower and upper lines of the whisker), and outliers observations ("+" signs), if any. This representation is helpful to visualise and analyse the data distribution.

In the following figures, box-and-whisker plots are used to present the summarized data measurement of four position trackers at three points on the 3D grid. These points are the origin [0, 0, 0], [-1; -1; 1] (normalized), and [2; 2; 2] (normalized), and were selected to show the position tracker performance in the edges of the common space. In these plots we used actual values in cm, with a blue line across the plot as the nominal position of the measurement. Also, a light-blue bar is plotted just to visualise the amount of error.

Figure 4.13 shows the data reported by the four trackers at the origin of the system on the x, y, and z axis. The blue line and the value at the right of each graph, represent the nominal position of the measured point. Overall, the system with best accuracy and precision performance is the Vicon, with median values along the axes close to 0 and little variability of the measurements. This high precision is shown by the small size of the boxes and whiskers for the Vicon data. The Polhemus also shows good performance in terms of accuracy, but the blue lines in the three axes representing Q1 and Q3, and the smaller black line in z indicates that the distribution of the reported data is more sparse than the Vicon, meaning that it is less precise. The Gametrak plot shows median values very close to the Polhemus, but with less precision, especially in the z-axis. The Kinect is the least accurate and precise of all trackers, reporting a median value for x and y with offsets close to 17 cm and 10 cm, respectively.

Figure 4.14 shows reported data for the point located at the normalized point in space [-1; -1; 1]. The blue line and the value at the right of each graph, represent the nominal, actual position of the measured point. The Vicon system was the most accurate and

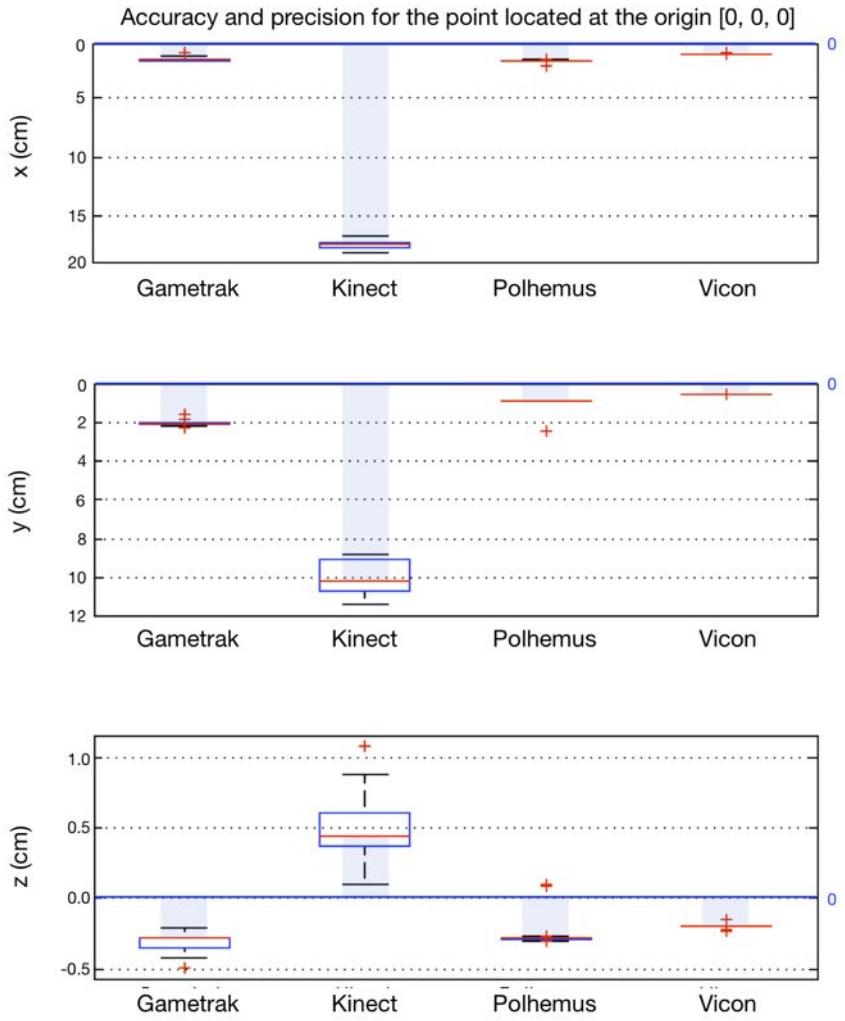


Fig. 4.13 Reported data by the four trackers at the origin [0, 0, 0]

precise of all trackers, with its reported data very close to the nominal position and little variability. The Polhemus followed the Vicon, showing its data focused close to the median, meaning precise measurements. However, it showed a difference in accuracy of 7 cm in the y axis. The Gametrak measurements for this specific point were as precise as the Polhemus, but it showed less accuracy due to the scaling-distance dependant factor that affects its measurements. The Kinect shows accuracy errors of similar magnitude, in the range of 5 cm and 10 cm, for the three axis. Again, the Kinect was the least precise of all compared trackers.

Figure 4.15 shows the reported data for the point located at [2; 2; 2]. The blue line and the value at the right of each graph, represent the nominal, actual position of the measured point. As can be seen in the plot, the Vicon system was again the most accurate and precise of all trackers, and the three other systems showed similar magnitude of inaccuracy but in different proportions for each axis.

4.3.3 Data measurement rate

For comparing the data measurement rate of the trackers, we subtracted the time-stamps on two consecutive measurements, over all measurements. By doing so, we could observe how constant in time each tracker reported its data, as well as the average rate of the measurements. As mentioned before, it is important to keep in mind that these values did not represent the update rate of each tracker alone, but the whole chain of processes and tools we used. Figure 4.16 shows box and whisker plots for the update rate for each system. The left plot provides a general view of the data for all systems and the right one is a scaled version of the Gametrak, Kinect, and Vicon data.

The Vicon system reported spatial position with a median rate of 100Hz. Half of the reported values were sent at rates between 95Hz and 112Hz, and the lower and upper adjacent of the distribution are at 71Hz and 125Hz. However, there are 22 outlier data points which are not considered part of any quartile because they are outside of the distribution curve. Analyzing the raw data, we realized that some of the measurements were sent after longer times than the average update rate, but the next ones came very close to the previous one. Also, in the Vicon software we selected an update rate of 240Hz, which clearly is much larger than what we obtained in our experiment. Because of these two issues, we speculated that perhaps there was a bottleneck in some part of the data flow for the Vicon

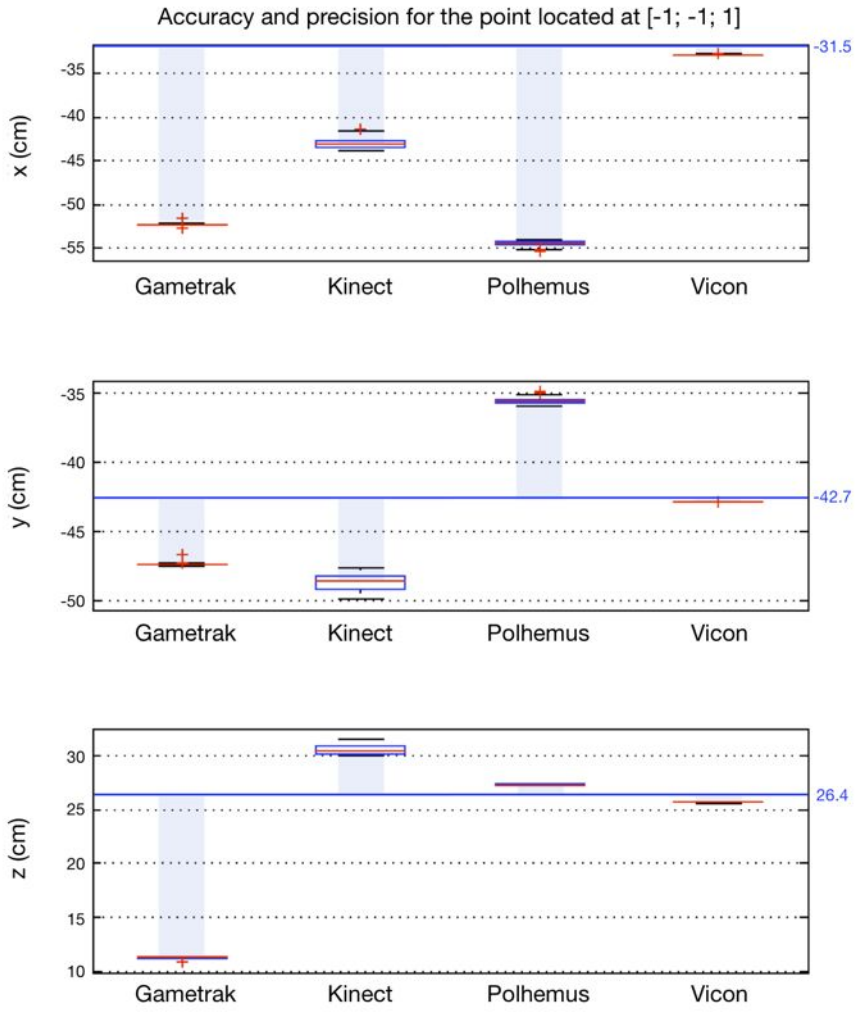


Fig. 4.14 Reported data by the four trackers at [-1; -1; 1]

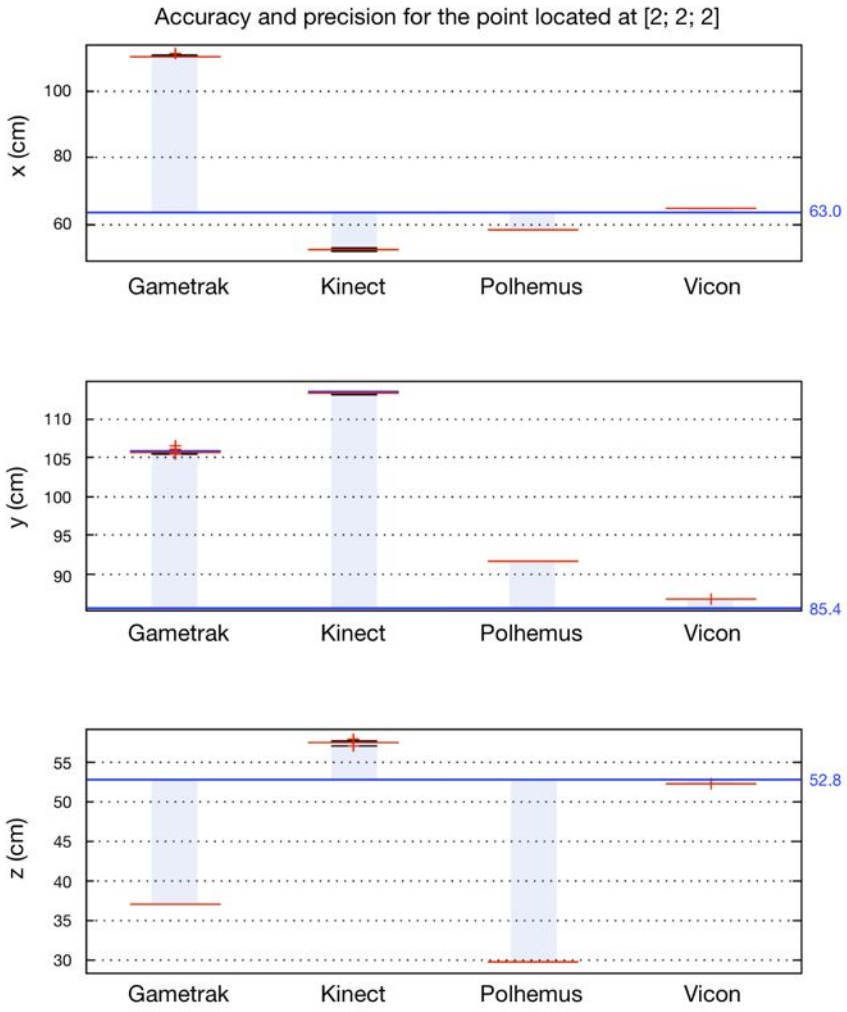


Fig. 4.15 Reported data by the four trackers at [2; 2; 2]

system. As we used almost the same pipeline for all systems, and we did not find this kind of bottleneck with the others, the problem could be in the different parts of the data flow. In the case of the Vicon, this stage corresponded to the QVicon2OSC application, which we do not have a way to monitor the signals it parses and processes.

The Kinect output median rate is 31Hz, and it is the more stable over time of the trackers we tested. Half of the samples were sent between 29.2Hz and 31.3Hz and there are no outliers. However, the sample rate is very low in comparison with the other systems, and that could help to make the data flow constant.

The Gametrak shows a median update rate of 99.3Hz. However, the distribution is uneven, with the 75th percentile located at 112.9Hz and the 25th percentile at 48.03Hz, almost half the median update rate. Also, the values for the lower and higher adjacent, extreme values of the distribution without considering the outliers, are located at 13.42Hz and 203.27Hz. This large deviation of the data distribution means that the Gametrak reported its data in a very unsteady way over time.

The case of the Polhemus is different. In our testings, this tracker reported an average of 236 points per second which is almost the actual specification for the system (240Hz). However, when we analysed the arrival time of the data, we noticed that it arrived in a very uneven form. Although the Polhemus is reporting the correct number of points according to its specifications and settings, the distribution of the arrival time of these values is extremely sparse. We analysed the recorded data and observed that for every 25 values with a short time of arrival, there was a much longer one. This sequence was repeated in every measurement for the 80 points on the grid. To isolate and analyse this issue, we did not send the values to the other computer and used the *plhm* software capabilities to record the data internally. We obtained similar results to those when we sent OSC messages to another computer. Hence, in order to analyse only the raw values generated by the Polhemus, without any other further processing of the data, we used *PiMgr*, the Polhemus' Windows GUI application, to write the output of the tracker information data to a text file (PiMgr does not allow to parse and stream raw Polhemus data to another application or computer).

Figure 4.17 shows the plot of the data retrieved using the PiMgr application. It shows the difference in the arrival time between two consecutive reported data points over a lapse of 15 seconds. The median of the time differences is 4ms. However, every 61.5 values (or 256.25ms), there is delay of 20ms in the next reported point which is compensated later

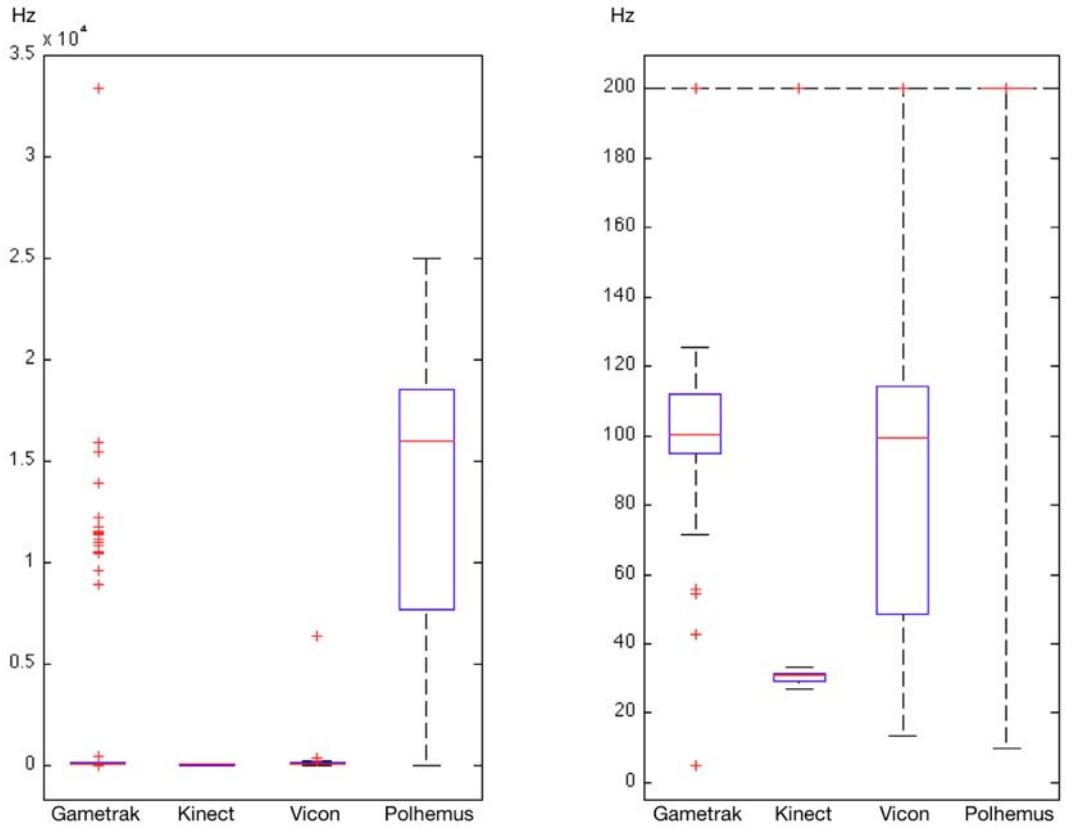


Fig. 4.16 Update rate for the four position trackers

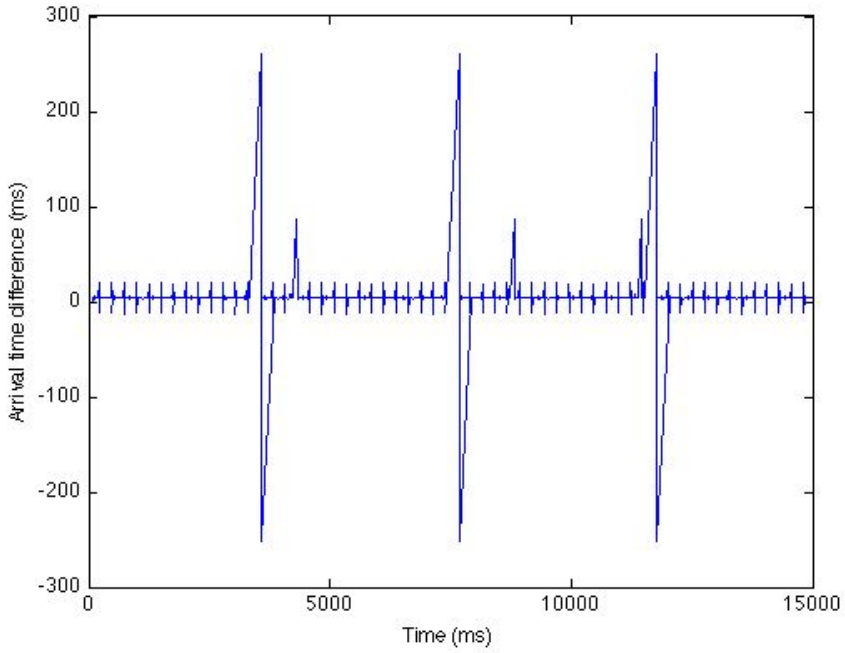


Fig. 4.17 Arrival time difference of the Polhemus data

(the small peaks). Every 781 measurements (or 3500ms), a larger correction, in the order of 260ms, is performed by the system. There are also three peaks that are not corrected (circa 4200ms, 8500ms, and 12000ms).

This behaviour of Polhemus Liberty is very interesting. The system is reporting the number of points per second that it should report, but in an uneven form. We reviewed again the Polhemus documentation and It is intriguing that some documents state that the system can provide 240 measurements in one second, per sensor, but other documents state that the update rate of the system is 240Hz. Both statements are different, while we confirmed that the former is true, we proved that the latter is false.

4.4 Summary

This chapter has presented an experimental comparison of four position trackers systems. Some of their performance parameters have been measured in the same environment with similar conditions, and a common workflow pipeline has been used to compare their reported data.

Overall, the tracker with the best performance is the Vicon 460 Motion Capture system. The shape of the space it senses is the closest to the nominal space, and is the most precise and accurate of all tested trackers. In terms of its update rate, however, the workflow pipeline we used with the Vicon provides less than half of the rate we expected according to our settings of the system. Still, this output rate is good enough to track the kind of gestures we use in our system.

The Kinect tracker lacks of precision and accuracy in all zones of the space. However, the shape of the space it senses is very close to the nominal space. The workflow pipeline we used in companion with the Kinect has the most stable output rate of all systems, but at the same time is the slowest one. It is, however, the system with the simplest set up and calibration processes, and it is immune to changes in lighting conditions.

The measurements of the Polhemus are very biased by the presence of metallic surfaces and objects in the room that we were not aware. Because of these conditions, the space it reports is close to the actual space, but only in a limited region, close to the source. Beyond a certain limit, the system reports the position of the points very biased toward the ceiling, floor, and walls. In terms of its output rate, the Polhemus reports values at an uneven rate, but it compensates this shifts in time to meet its declared specifications.

Finally, the Gametrak mechanical tracker reports a curved space. Values for points in the vicinity of the device are close to their actual position, but distant points are gradually deviated from their nominal position in the space, following the path of curved, concentric lines with their origin in the device. The time of arrival of the Gametrak measurements is reasonably constant and fast for the requirements of our system.

During this research, we have found that the design of a touchless musical interfaces can be delineate by the performance parameters and practical characteristics and considerations, which can affect its use on-stage, of a position tracking system. For this reason, to choose the most appropriate system for tracking the musical gestures of a performer in performance contexts, a compromise between its two set of parameters, technical and practical, is necessary.

For our implementation, we have chosen to work with the Kinect position tracker system. On the one hand, the shape of the space it senses is close to the actual space. This factor is perceptually relevant because the Kinect allows for a linear relation between the performer's movements and position in the performance space, and the acquisition and mapping of these values to the descriptor space. On the other hand, the portability of the Kinect,

its fast set up, easy calibration process, and its immunity to lighting conditions (which is especially relevant for performance contexts, outside of the controlled conditions of a laboratory), offers us the best trade-off between the performance parameters and practical characteristics of the position trackers we tested.

Chapter 5 will present the implementation we designed in the creation of a touchless gestural interface to control a concatenative sound synthesis system.

Chapter 5

Immersed in Sounds: *SoundCloud*

One of the main goals of this thesis is to develop a system to browse, perform and compose with sound units of a sonic database arranged in a 3D descriptor space by means of open-air gestures.

To achieve this goal, in previous chapters we reviewed some of the technologies for non-contact spatial tracking, and we showed examples of musical interfaces and instruments developed using these tracking techniques. We also empirically compared four professional and consumer-oriented trackers according to some of their performance parameters, and we showed some of the strengths and limitations that these systems have. Finally, we compared and summarized the characteristics of three different CSS implementations, and showed some examples of pieces and implementations using the interaction possibilities these systems offer.

It is in the spirit of this research that the implementation we propose will increase the control and will improve the expressiveness of CSS by allowing simultaneous access to low- and high-level sound descriptors. We also expect that the visual link between the performer's gestures and the sounds that these movements generate with our system will increase the audience's engagingness of looking at an untethered performance. As we plan to use the proposed system for performance contexts, its implementation ought to be portable, easy to set up and calibrate, and should not be affected by the lighting conditions and metallic surfaces and objects in the space it will be used.

5.1 *SoundCloud*

The system we have developed is called *SoundCloud*. In SoundCloud, a performer creates or imports one or two sound corpuses of arbitrary sonic sources, and selects in the computer screen what descriptors are used on each one of the performance space axes, thus obtaining subspaces of the higher-dimensional spaces of the sound corpuses. The values of the descriptors for each one of the sound units are normalized beforehand in order to limit and constraint them to the same performance space.

The sound corpuses as well as the chosen descriptors can be set up to play the system in four different combinations:

- **One sound corpus and one set of descriptors** is the simplest representation. Sound units are distributed in the performance space according to the chosen descriptors for each axes. This configuration gives the performer a clear idea of how the units are distributed in the space, and provides clues about the practical meaning of the descriptors.

- **Two sound corpuses in one descriptor space** allows the performer to play with two databases of sounds distributed in the same descriptor space. Each hand plays a different sound corpus, so locating the hands in the same position in the space will play different sounds but with similar characteristics according to the chosen descriptors.

- **One sound corpus in two descriptor spaces** allows the user to play the same units in two different parts of the performance space. This configuration requires that the user loads the same sound corpus for each hand.

- **Two sound corpuses in two descriptor spaces** is the most complex of the four configurations. Two databases of sounds are loaded and each hand is mapped to an unique sound corpus. The descriptors for each hand could not be correlated, so positioning both hands in the same point in space plays two sound units with different sonic identities.

In order to make the system easier to play with, SoundCloud provides visual feedback by plotting the sound units in a virtual three-dimensional space. This feature allows the

performer to see where the sound units are located in the performance space, guiding her in the exploration of this descriptor space. This characteristic is useful because it is very common that the unit's distribution in the space is uneven (Schwarz et al. 2006), thus generating dense clusters of sound units in some zones of the space, sound clouds, and leaving other zones almost empty or with units sparsely distributed, which is not optimal for performance. Also, the visualisation tool allows the performer to see her hands plotted in the descriptor space and to control the camera view position according to the position of her head. Thus, walking backward, for instance, will allow the performer to have a general, broad view of the descriptor space, and walking forward will provide her with a closer view of the sound units close to the camera view position.

5.2 Design and Implementation

In order to achieve expert interaction by means of the use of gestural input devices to control real-time sound synthesis and to give the required importance to all design stages of an interface for musical expression, it has been suggested that the process of development of such a system should be divided in four parts (Wanderley and Depalle 1999):

- Definition and typologies of gesture
- Gesture acquisition and input device design
- Mapping of gestural variables to synthesis variables
- Synthesis algorithms

For the SoundCloud design and implementation, we added a fifth stage related to provide visual feedback to the user, thus improving the playability and degrees of expertise able to achieve with our system.

Figure 5.1 shows a schematic diagram with the main blocks of SoundCloud. Gestures are represented by the performer controlling the hands and head position at the left of the figure. The gesture acquisition and input device stage is coloured yellow, the mapping stage is green, the synthesis part is blue, and the visualisation pink. Lines represent the control signals flow between the modules.

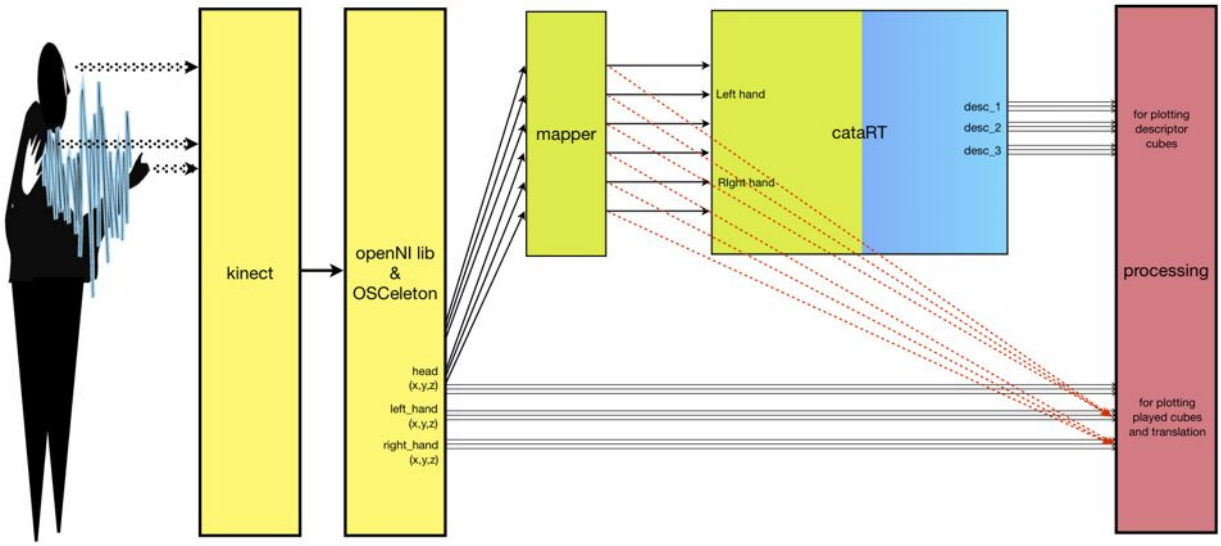


Fig. 5.1 SoundCloud implementation schematic diagram

Definition and Typologies of Gesture

In SoundCloud, we defined a simple interaction metaphor to explore and play the sound units located in the descriptor space. Although the performance space could be a multi-dimensional scaling of a higher-dimensional similarity space, we opted to simply give the user the possibility to choose and assign a unique descriptor for each axis, making it easier for her to create a meaningful performance space. Thus, each one of the performer's hands is considered as a playback head that activates and reproduces the closest sound unit in the performance space. This performance space is mapped *one-to-one* with the descriptor space, so each one of the positional axes, x, y, or z, is directly mapped to a chosen descriptor axis. This simple, low-level mapping was chosen in order to provide the performer with a direct access to the selection of the sound units in a user-defined subspace of the multi-dimensional feature space of the sound corpus. A higher-level, *one-to-many* mapping could be implemented, but the relation between the performance and the descriptor spaces would be not as clear as with our approach. Hence, the user can explore the sound corpus' units, located in the performance space according to their descriptor values, by moving and positioning her hands in some parts of the space, thus triggering the closest sound unit and chaining it with the previous one according to the chosen chaining mode. The selection of the descriptors for the axes is not fixed, allowing the performer to choose which set fits

better with different sound corpuses or performances. Having said that, in our preliminary testings the combination of spectral centroid, note pitch, and loudness for x, y, and z respectively, was very effective to provide a meaningful distribution of the sound units in the performance space.

Data-driven CSS usually requires an audio input to drive it, but the SoundCloud design does not need an audio input because it allows to select and play the sound units by means of user-driven gestures in the air. Thus, using gestures to control which sound units from the sound corpus will be played back allows the user to explore and perform with a sound corpus at will, broadening the range of sonic possibilities because all zones of the descriptor space, even its boundaries, can be easily reached. Our gesture-based approach also gives the performer, as well as the audience, of feedback because the position of the performer's hands in space can be related to specific sounds of the sound corpus, creating a deterministic performance. On the other hand, if the performance space is large, it could be hard for the performer to play sound units located in opposite sides of space.

Gesture Acquisition and Input Device Design

In Chapter 2 we reviewed different technologies and systems that can be used to acquire the performer's gestures, and some interfaces that have implemented that tracking technology for musical composition and performance. In Chapter 4 we empirically compared four tracking systems according to their characteristics and performance parameters. This review and comparison of tracking systems gave us insights for the development of SoundCloud. According to the experimental results, the best choice would be to use the Vicon 460 motion capture system. However, because of its inherently complex characteristics of calibration and set up, its use would be most of the time confined to a laboratory setting, limiting its use in performance contexts. The factors of portability, easiness of calibration and set up, as well as the capability to work with almost any lighting conditions, made we choose to work with the Microsoft Kinect position tracker.

Although in Chapter 4 we demonstrated that the Kinect is the less precise and accurate of all systems we compared, the shape and dimensions of the space it senses (see Figure 4.10) is close to the user's mental representation of a 3D space with orthogonal axes. In other words, there is a direct, linear mapping between the performer space sensed by the tracker and the descriptor space. Thus, by using the Kinect we provide the performer with

a more clear mapping between her gestures and the playback of the sound units in the sound corpuses, although at the expense of lower accuracy and precision. Interestingly, we also considered that in some circumstances the lack of precision of the Kinect could improve SoundCloud musicality by allowing to stochastically play units with similar descriptors around the nominal position of the hands of the performer, helping the unit transformation stage and improving the concatenation. Furthermore, we informally tested the Polhemus and the Gametrak systems in SoundCloud, and experienced that the mapping between the two spaces, performance and descriptor, was not as clear as with the Kinect. This issue is due to the curved space these trackers sense in the test environment (see Figures 4.11 and 4.12).

Mapping of Gestural Variables to Synthesis Variables

In Chapter 3 we reviewed three different systems that implement CSS. After comparing some of their features and the possibilities they provide in the context of our project, we decided to use CataRT due to its capability to work doing data- as well as user-driven CSS, extensive documentation, batch-processing possibilities, variety of descriptors, the calculation of the characteristic values of the evolving descriptors, and its modular, open architecture using Max/MSP.

The mapping of the gestural variables in SoundCloud is done in two stages. First, libmapper and the mapperGUI are used to select and map the values for the user's position acquired with the Kinect. Libmapper also provides the ability to scale all signals, so we normalize the positional data to a range of [-1.,1.] in order to have a common scale in the values of the three axis for the position of the hands and head.

In a second stage of the mapping, the user can select in CataRT the descriptors mapped to each axis for each hand. By doing this mapping, the performer is explicitly declaring the descriptor space that she wants to use for the sound corpus and how her hands positions in the performance space are mapped to the descriptor space.

As the range of values for different descriptors is commonly different, SoundCloud takes care of dynamically finding the maximum and minimum values according to the descriptors selected by the musician and use those values to scale and normalize the descriptor space to the performance space. Although the actual value of the descriptors is never changed, this scaling of the values is important for the visualisation stage.

To achieve a fast access to previously stored settings of descriptors, a set of memories is implemented in the main control patch of SoundCloud. By clicking the memory buttons in the computer screen before the performance, all descriptors and their minimum and maximum values are loaded in order to scale their values. Also, after triggering a memory each of the sound unit descriptor values is sent to the visualisation tool to plot the sound corpus according to the new axes in the descriptor space.

Figure 5.2 shows the SoundCloud implementation based on CataRT. Green and light-orange boxes allow for the mapping of each one of the axis of the hands positions to a user arbitrarily-chosen descriptor. Lower-left grey box receives the scaled values from libmapper for each hand and the head position and maps it to the unit selection in the green and light-orange boxes.

5.3 Synthesis Algorithms

All the synthesis stage in SoundCloud is performed by the CataRT synthesis module. Its GUI also offers access to the following methods for doing transformation of the sound units: *transposition*, *gain*, *reverse*, and *panoramic distribution* of the sound in a stereo output. Values for each one of these variables can be set up before the performance using the computer keyboard. These methods are very useful to improve the sound unit concatenation since CataRT does not consider the concatenation distance between two selected units adjacent in time. SoundCloud also considers in its implementation some of the different modes that CataRT offers for chaining the selected sound units: *bow*, *fence*, *beat*, *chain*, and *continue*. These trigger methods must be set up before the performance, and allow the user to control how the sound units will be triggered. See subsection 3.2.2 for more details on each one of these methods. On the other hand, in its current implementation SoundCloud does not allow to control the crossfade fade-in and fade-out times offered by CataRT as well as their tools for granular-style synthesis or overlapping sound units in clouds.

5.4 Interface and Visualization

Improving the visualisation of the timbre-space where the sound units are located will also improve the way a user can interact with the sound corpus.

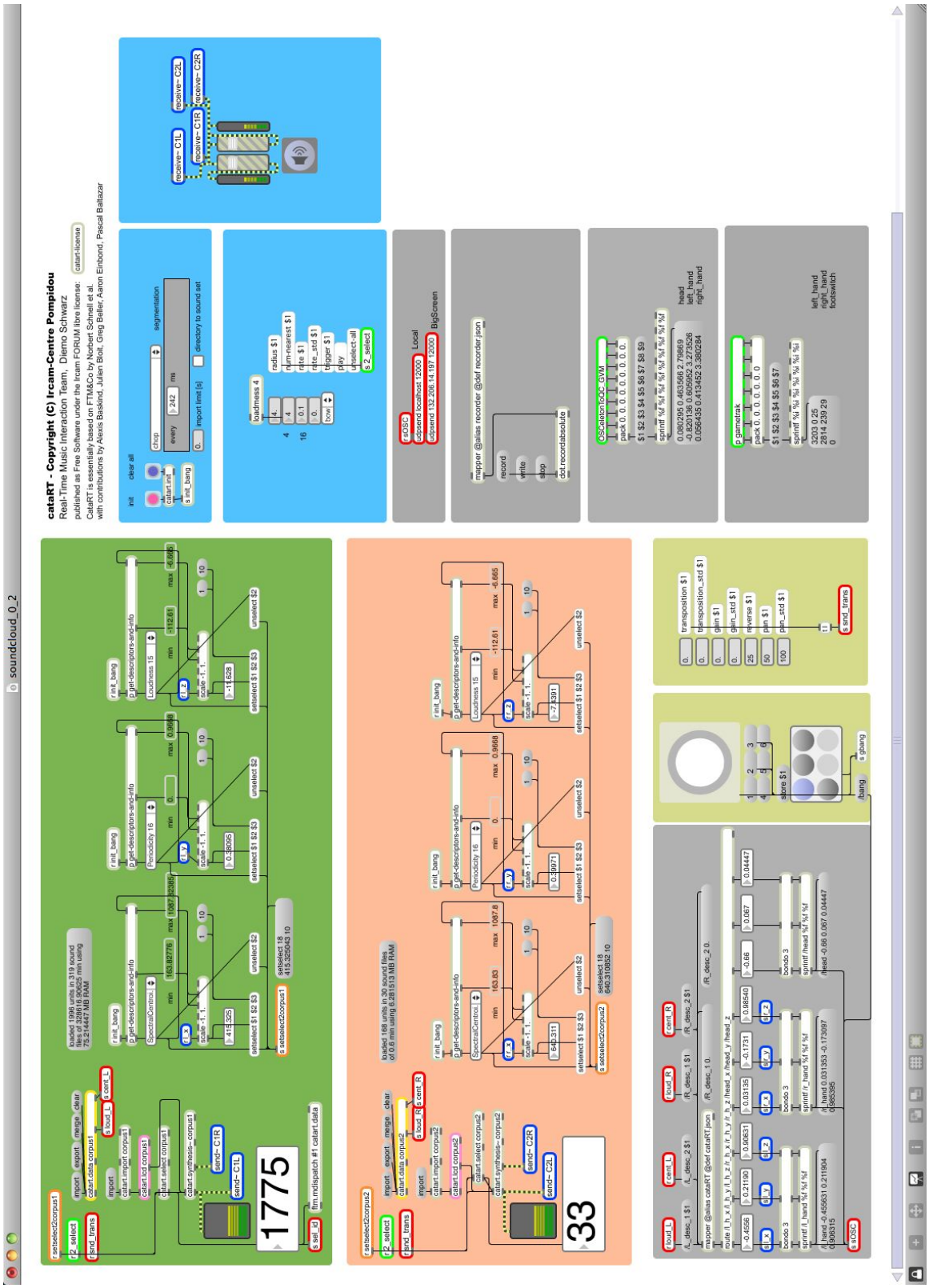


Fig. 5.2 SoundCloud Graphical User Interface

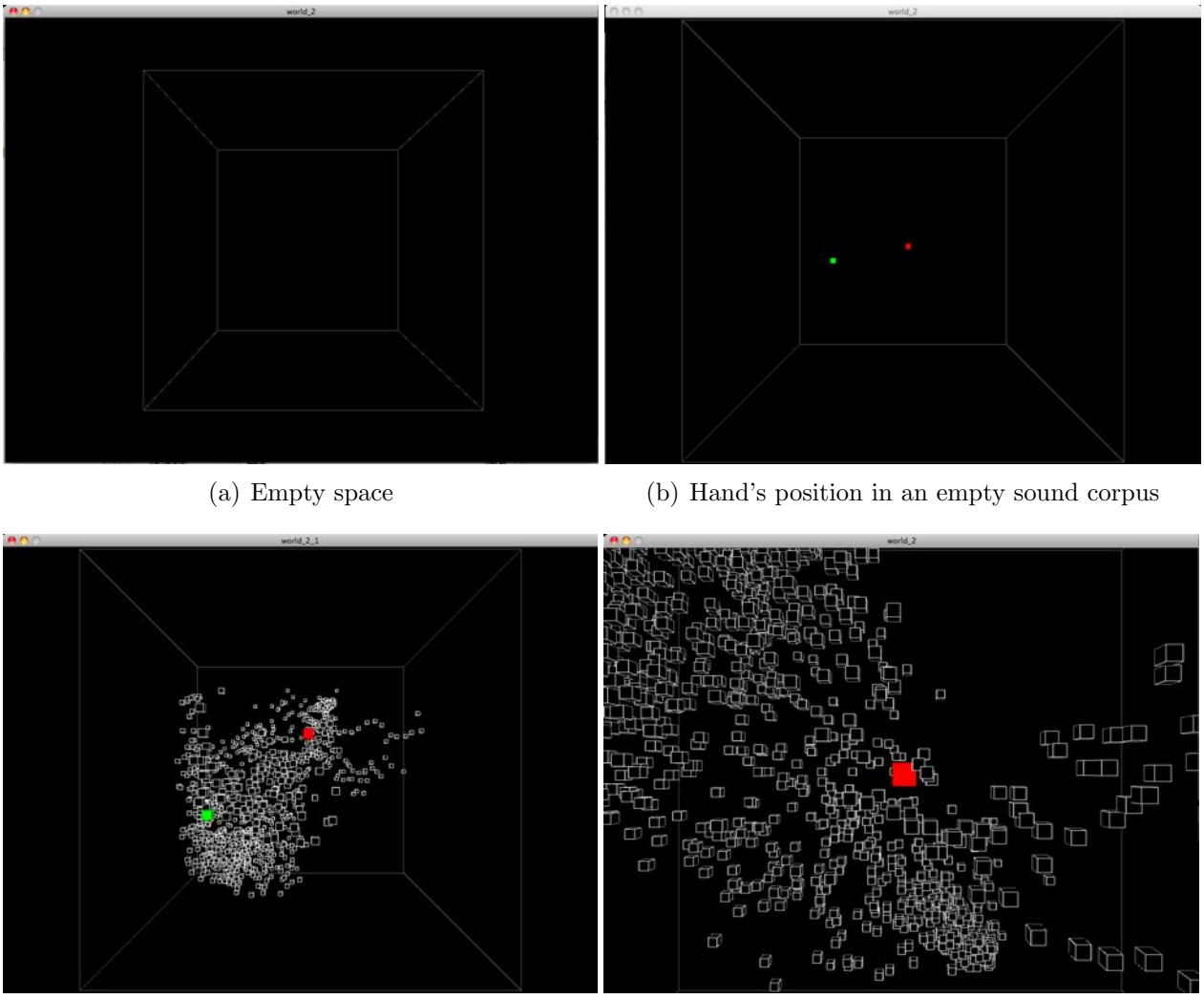
The SoundCloud GUI interface controls all aspects of setting up the SoundCloud environment for playing with it. When this is done, the interface for interacting and performing with the sound corpus is the performance space itself. This performance space has no physical boundaries (except maybe by the floor, walls and ceiling if our performance space is large enough) and the musician cannot know in advance where the sound units are located in space according to the chosen descriptors. This situation is particularly important in the stage of development of a sound corpus because although the performer may have some clues about where the sound units will be located and how they will be distributed in the performance space, their actual position will depend on the minimum and maximum values of all sound units for the selected descriptor. Hence, it is hard to know in advance where the sound units will be located and how they will be clustered. Once the sound corpus is designed and the descriptors for distributing the sound units in the space are chosen, the position of the sound units in the space will be fixed. This moment is crucial because the performer can learn proprioceptively, according to her own body, where the sound units are located.

To inform the performer about the representation of the sound corpus in the descriptor space, and help her with the control and interaction with it, our visualisation application should give the performer an idea of the boundaries of the descriptor space, where the sound units are located, and should change the camera position according to the position of the performer in the performance space.

We chose the Processing open-source programming language[1] for developing the SoundCloud visualisation application. Processing is especially well suited to work with images and interaction, and has an extensive list of third-party libraries. For the development of SoundCloud, we end up using the OpenGL library for creating accelerated 3D graphics.

Our Processing-based visualisation application is shown in Figure 5.3. Subfigure 5.3(a) shows the empty space with a plotted cube showing the boundaries of the descriptor space. Subfigure 5.3(b) shows the space without any sound corpus loaded, but showing the performer hands in the space. Each hand has its own colour, green for left, red for right. As can be seen, the camera position in the second slide is slightly closer to the cube than in the first one, denoting that the performer moved into the performance space. Subfigure 5.3(c) shows a sound corpus with 2000 sound units plotted in the space according to the chosen descriptors. As expected, the sound units are clustered in certain sectors, and large zones

[1] `http://processing.org/`, accessed May 18, 2011



(a) Empty space

(b) Hand's position in an empty sound corpus

(c) Playing sound units in the sound corpus

(d) The head position controls the space view

Fig. 5.3 SoundCloud space visualisation

without sound units exist what can be very positive in an immersive environment, allowing the performer to "scape" the interface, i.e., move without generating sounds (Mulder 2000). Coloured cubes indicate the performer's hand position. In subfigure 5.3(d) the performer moved forward. The spatial values of her head's position are used to change the camera view, providing a closer look to an specific zone of the space. This zoom feature is helpful for exploring clusters of sounds and see the overall distribution and shape of the sound corpus in the descriptor space.

5.5 Discussion

This section provides a discussion on each one of the stages of development of SoundCloud. It focuses on the insights we obtained after the design, development, and testing of our system.

Definition and Typology of Gestures

SoundCloud implements a system to explore and perform with a 3D sound corpus by means of a touchless interface. When the performer enters the sensed space, the position of her hands is acquired, these spatial values are mapped linearly to the descriptor space where the sound units are located, and the closest sound unit to each hand is played back. Hence, the performer is able to control simultaneously with each hand three descriptor values mapped directly from the descriptor space. Thus, SoundCloud efficiently extends the interaction possibilities of previous CSS implementations—commonly done in 2D descriptor spaces using a computer mouse and keyboard—allowing a performer to explore and perform with a sound corpus by means of 3D gestures in the performance space. This added dimension helps her to refine the exploration of a sound corpus because the sound units are less clustered and more evenly distributed than in 2D.

Gesture Acquisition and Input Device Design

SoundCloud's current implementation works well with the typology of gestures defined for the system. This implementation is based on tracking the hands and head of a performer using the Kinect position tracker. Although we empirically found that is not the most accurate or precise of the trackers we compared, we ended up using this device because

it offers the best trade-off between its performance parameters and the on-stage practical considerations of the trackers we tested, as well as a welcome variability in position measurement that allows for musically interesting effects.

Mapping of Gestural Variables to Synthesis Variables

SoundCloud uses libmapper to create flexible mappings between the performance and the descriptor spaces. This tool allows for rapid testing of new input or tracking devices, such as in the experimental comparison showed in Chapter 4, for routing the hands' spatial values in the performance space to control other processes, such as an spatialisation application, for instance; or for mapping the gestural variables to new sound units' descriptors. Expanding on this last idea, while new, perceptually meaningful descriptors are developed, SoundCloud can be used to continue experimenting with the currently implemented descriptors in CSS systems. In particular, CataRT has a powerful, not commonly used set of descriptors based on high-level, user annotated attributes. These descriptors allow to give the units a "unit type", for instance, creating subsets of sound units. These descriptors give more control to the performer, allowing her to create more complex mappings and meaningful sound corpuses.

Synthesis Algorithms

Although the sound synthesis engine of the CSS system is out of the scope of this thesis, its sonic output is crucial for a good perception and evaluation of the whole system. Because the concatenation of the sound units in real-time is still rough, we included some of the sonic transformations that CataRT provides, such as panning, reverse, and transposition, and we implemented time-based digital audio effects, such as delay and reverb, to create perceptually smoother transitions of the sound units.

Feedback

When creating new digital musical interfaces such as SoundCloud, instruments that by definition do not have previous methods of study or repertoire, the feedback these systems provide to the performer is crucial to allow her achieving an expert performance.

SoundCloud provides simple sonic feedback by playing back active sound units. The audio output of these sound units is monophonic and transformation processes can be used

to distribute the sounds in the stereo field.

The SoundCloud's visualisation application showed in subsection 5.4 allows a user to visually explore a sound corpus. The performer can also look at where her hands, the "playback heads" of the system, are located in the space. This visual representation is where the descriptor and performance spaces meet. In other words, the physical, concrete world of the performer and the non-tangible, abstract world of the descriptors converge in this representation. Furthermore, this dynamic representation of the sound corpus is explorable and zoomable by changing the camera view, which is controlled by the performer's head position. Because it is difficult for the performer to know in advance where the sound units of an unknown sound corpus will be located in the descriptor space or how these units will be distributed according to different descriptor schemes, the visualisation application provides a major channel of feedback for the performer and it is a key part of SoundCloud.

5.6 Summary

In this chapter, the musical requirements and design goals of SoundCloud were stated. The performer's gestures were chosen, and the input device for the gesture acquisition was defined. The open, user-choosable mappings between gestures in the performance space, and their link to the descriptor space in the CSS system were described. A diagram with all major sections of SoundCloud was shown, as well as the GUI to control all aspects of the system and the visualisation application. Finally, a discussion of the achievements of the system was presented, according to the design stages of a new interface for musical expression.

Chapter 6

Conclusions: Present and Future

This chapter summarizes the research presented in this thesis. It also discusses ideas, gives conclusions, and presents paths for future development on the topics covered in this work.

6.1 Conclusions

This thesis began by providing the basic concepts and characteristics of non-contact spatial tracking, reviewing different technologies to achieve this objective in the context of musical instrument design, and surveying musical interfaces developed using these systems.

The "perfect" position tracker system or technology does not exist. A specific system can work well in a certain context and for certain uses, but can not work in others. At the same time, diverse technologies can be used to track the same gesture, providing different results according to the performance parameters, that is the intrinsic characteristics of a particular system. Considering these parameters is key to select one specific system among many possibilities, so we empirically compared four position tracking systems based on different technologies. A workflow pipeline was designed to extract and analyze the data that each one of the trackers measured, and the performance parameters of these systems were calculated. At the same time, as we are involved in the design of musical instruments to be used in performance contexts, a complementary set of characteristics particular to each system, but related to the variability of the environments for musical performance, was considered in the analysis because it could inform and modulate our decisions over the design. We decided that a trade-off between the optimal technical parameters and the various characteristics that make different systems more or less suitable for performance

in different contexts ought to be considered in order to design a new touchless musical interface.

The results of the experimental comparison revealed some of the strengths and weaknesses of the position tracking systems and their workflow pipeline in the context of the experiment. Although the Kinect was not the most accurate and precise of all trackers we compared, we ended up choosing it because it offered the best compromise of its characteristics for the context of our design.

As one of the goals of this thesis is to perform with sound units taken from a database of sounds, we discussed CSS and three implementations of this synthesis technique were reviewed. Their stages of analysis, selection, and synthesis were compared, and an overview of their visualisation and interaction possibilities was shown. For our research, we chose CataRT as the CSS application to work with because of the balance of features it provides in terms of sound segmentation possibilities, nature of descriptors, open architecture and extensibility, and extensive documentation.

Finally, we integrated all previous insights in the design of SoundCloud, a touchless gestural interface to control a CSS system. Our design allows a performer to control, by means of the position of her hands in a 3D performance space, two target points in a 3D descriptor space using one or two sound corpuses. In order to facilitate the exploration of these sound corpuses, we also developed a system to visualise their representation in the descriptor spaces.

6.2 Future Work

Gestures

The typology of gestures used in SoundCloud is fairly simple, the position of the performer's hand, two points in the performance space mapped linearly to the descriptor space. Future research should consider the development of a set of more complex gestures to generate and activate processes in the performance space, such as to create and activate loops of sound units, "freeze" sound units or processes, or create *playback paths*, for instance. Other gestures could also be used to switch from *performance* to *control* mode, allowing the user to control aspects of the SoundCloud GUI, such as the selection of descriptors, activating the sonic transformations, or zooming to specific, more dense zones in the descriptor space,

for instance. Extending this idea of creating new performance and control gestures, sounds produced by the performer's voice could also be used to control certain descriptors, for instance the spectral centroid, which is commonly used as one of the descriptors and can be easy to control to a certain degree with vocal sounds, or to trigger processes, such as activating or de-activating the system. By doing this, we could integrate another dimension of control to SoundCloud.

However, if new gestures are planned, another study should be considered to define to best device for the gestural acquisition. This other device ought to be capable to track user-choosable, well-defined points in the human body. By doing this, shapes could be created by linking several points, and the rotation of these shapes could be calculated. Thus, a 6DOF system could be used to control a more complex set of gestures controlling the SoundCloud processes and GUI.

Immersiveness

Future work in SoundCloud should also consider improving the immersiveness of the system by means of the spatialisation of the sound units according to their absolute position in the descriptor space, and the creation of a cave-like, 3D graphic representation of the space populated with the sound units. Thus, the performer could explore the sound corpus being immersed, visually and acoustically, into the sound units. This improvement would require to optimize the Processing code, or better, to write a lower-level implementation of the visualisation application. This new coding should allow to plot (and rotate, and translate in real-time) sound units of larger sound corpus. Moreover, the libmapper's Java bindings should be implemented in the visualisation application in order to go straight from one application to the other, thus saving computational cycles.

Future Uses

Finally, a qualitative evaluation of the proposed system should be done in order to assess what stages could be improved. Also, research using longer audio units, such as sonic clips or songs, could be done in order to evaluate the feasibility of using SoundCloud for exploring sound files in audio effects collections, or for the creation of an interactive 3D music browser. We have now the tools for 3D music exploration, it is time to start playing.

Bibliography

Ascension Technology Corporation. 2011. Ascension technology corporation / flock of birds. Last accessed March 21, 2011, `http://www.ascension-tech.com/realtime/RTflockofBIRDS.php`.

Aylward, R., and J. Paradiso. 2006. Sensemble: a wireless, compact, multi-user sensor system for interactive dance. In *Proceedings of the 2006 Conference on New Interfaces for Musical Expression*, Paris, France, 134–9.

Bongers, B. 2000. Physical interfaces in the electronic arts. In M. M. Wanderley and M. Battier (Eds.), *Trends in Gestural Control of Music*, 41–70. Paris, France: IRCAM.

Brent, W. 2009. Cepstral analysis tools for percussive timbre identification. In *Proceedings of the 3rd International Pure Data Convention*, Sao Paulo, Brazil.

Brent, W. 2010. A timbre analysis and classification toolkit for pure data. In *Proceedings of the International Computer Music Conference*, New York, NY, USA, 224–9.

Brent, W. 2011. timbreID, bfcc~ & other spectral features. Last accessed May 11, 2011, `http://williambrent.conflations.com/pages/research.html`.

Burdea, G., and P. Coiffet. 2003. *Virtual reality technology*. Hoboken, NJ, USA: John Wiley and Sons Inc.

Camurri, A., S. Hashimoto, M. Ricchetti, A. Ricci, K. Suzuki, R. Trocca, and G. Volpe. 2000. Eyesweb: Toward gesture and affect recognition in interactive dance and music systems. *Computer Music Journal* 24 (1): 57–69.

Camurri, A., B. Mazzarino, M. Ricchetti, R. Timmers, and G. Volpe. 2004. Multimodal analysis of expressive gesture in music and dance performances. In A. Camurri and G. Volpe (Eds.), *Gesture-Based Communication in Human-Computer Interaction*, Chapter 3, 357–8. Berlin, Germany: Springer Berlin / Heidelberg.

Carmody, T. 2011. How motion detection works in xbox kinect. Last accessed May 11, 2011, `http://www.wired.com/gadgetlab/2010/11/tonights-release-xbox-kinect-how-does-it-work/all/1`.

Casey, M. 2009. Soundspotting: a new kind of process? In R. Dean (Ed.), *The Oxford Handbook of Computer Music*, 421–53. New York, NY, USA: Oxford University Press.

Casey, M., and M. Grierson. 2007. Soundspotter/remix-tv: fast approximate matching for audio and video performance. In *Proceedings of the International Computer Music Conference*, New Orleans, LA, USA.

Chabot, X. 1990. Gesture interfaces and a software toolkit for performance with electronics. *Computer Music Journal* 14 (2): 15–27.

Chadabe, J. 1984. Interactive composing: an overview. *Computer Music Journal* 8 (1): 22–7.

Chadabe, J. 1985. Interactive music composition and performance system. Patent US4526078.

Couturier, J., and D. Arfib. 2003. Pointing fingers: using multiple direct interactions with visual objects to perform music. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*, Montréal, QC, Canada, 184–7.

Demeyer, T. 1996. BigEye (version 1.10). Last accessed March 28, 2011, `http://www.steim.org/steim/bigeye.html`.

Dobrian, C., and F. Bevilacqua. 2003. Gestural control of music: using the vicon 8 motion capture system. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*, Montréal, QC, Canada, 161–3.

Eckel, G., D. Pirro, and G. Sharma. 2009. Motion-enabled live electronics. In *Proceedings of the 6th Sound and Music Computing Conference*, Porto, Portugal.

Freed, A., D. McCutchen, A. Schmeder, A. Skriver, D. Hansen, W. Burleson, C. Nørgaard, and A. Mesker. 2009. Musical applications and design techniques for the gametrak tethered spatial position controller. In *Proceedings of 6th Sound and Music Computing Conference*, Porto, Portugal, 189–94.

Freedman, B., A. Shpunt, M. Machline, and Y. Ariely. 2010. Depth mapping using projected patterns. Patent US2010/0118123 A1.

Glinsky, A. 2000. *Theremin: ether music and espionage*. University of Illinois Press.

Hagedorn, J., S. Satterfield, J. Kelso, W. Austin, J. Terrill, and A. Peskin. 2007. Correction of location and orientation errors in electromagnetic motion tracking. *Presence: Teleoperators and Virtual Environments* 16 (4): 352–66.

Hasan, L., N. Yu, and J. Paradiso. 2002. The termenova: a hybrid free-gesture interface. In *Proceedings of the 2002 Conference on New Interfaces for Musical Expression*, Dublin, Ireland.

Jacquemin, C., R. Ajaj, R. Cahen, Y. Olivier, and D. Schwarz. 2007. Plumage: design d'une interface 3D pour le parcours d'échantillons sonores granularisés. In *Proceedings*

of the 19th International Conference of the Association Francophone d'Interaction Homme-Machine, Paris, France, 71–4.

Jensenius, A., and V. Johnson. 2010. A video based analysis system for realtime control of concatenative sound synthesis and spatialisation. In *Proceedings of the Norwegian Artificial Intelligence Symposium*, Gjovik, Norway.

Joystiq. 2011. Kinect: The company behind the tech explains how it works. Last accessed May 11, 2011, `http://www.joystiq.com/2010/06/19/kinect-how-it-works-from-the-company-behind-the-tech/`.

Kindratenko, V. 2001. A comparison of the accuracy of an electromagnetic and a hybrid ultrasound-inertia position tracking system. *Presence: Teleoperators & Virtual Environments* 10 (6): 657–663.

Leslie, G., B. Zamborlin, P. Jodlowski, and N. Schnell. 2010. Grainstick: A collaborative, interactive sound installation. In *Proceedings of the International Computer Music Conference*, New York, NY, USA, 123–6.

Lima, G. H. T., M. Maes, M. Bonfim, M. V. Lamar, and M. M. Wanderley. 1996. Dance-music interface based on ultrasound sensors and computers. In *Proceedings of the 3rd Brazilian Symposium on Computer Music*, Recife, Brazil, 12–6.

Mäki-Patola, T., J. Laitinen, A. Kanerva, and T. Takala. 2004. Human model interaction: Report on final implementation and results. Deliverable D15, Laboratory of Telecommunications Software and Multimedia Helsinki University of Technology, Helsinki, Finland.

Mäki-Patola, T., J. Laitinen, A. Kanerva, and T. Takala. 2005. Experiments with virtual reality instruments. In *Proceedings of the 2005 Conference on New Interfaces for Musical Expression*, Vancouver, BC, Canada, 11–6.

Malloch, J. 2008. A consort of gestural musical controllers: Design, construction, and performance. Master's thesis, McGill University, Montréal, QC, Canada.

Malloch, J., S. Sinclair, and M. Schumacher. 2011. Digital orchestra toolbox. Last accessed May 1, 2011, `http://www.idmil.org/software/digital_orchestra_toolbox`.

Mathews, M. 1990. Three dimensional baton and gesture sensor. US Patent. 4,980,519.

Mulder, A. 1998. *Design of virtual three-dimensional instruments for sound control*. Ph. D. thesis, Simon Fraser University, Vancouver, BC, Canada.

Mulder, A. 2000. Towards a choice of gestural constraints for instrumental performers. In M. M. Wanderley and M. Battier (Eds.), *Trends in Gestural Control of Music*, 315–335. Paris, France.

Myers, E. E. 2002. A transducer for detecting the position of a mobile unit. Patent GB 2373039 A.

Paine, G. 2004. Gesture and musical interaction: interactive engagement through dynamic morphology. In *Proceedings of the 2004 Conference on New Interfaces for Musical Expression*, Hamamatsu, Japan, 80–6.

Paradiso, J. 1994. Penn and Teller seance electronics. Technical report, MIT Media Laboratory.

Paradiso, J. 1997. Electronic music: new ways to play. *Spectrum, IEEE* 34 (12): 18–30.

Paradiso, J., and N. Gershenfeld. 1997. Musical applications of electric field sensing. *Computer Music Journal* 21 (2): 69–89.

Piringer, J. 2001. Elektronische musik und interaktivität: Prinzipien, konzepte, anwendungen. Master's thesis, Institut für Gestaltungs und Wirkungsforschung der Technischen Universität Wien, Vienna, Austria.

PrimeSense. 2011. Primesense, reference design. Last accessed 11, `http://www.primesense.com/?p=514`.

Roads, C. 2001. *Microsound*. Cambridge, MA, USA: The MIT Press.

Rodet, X., P. Depalle, and G. Poirot. 1988. Diphone sound synthesis based on spectral envelopes and harmonic/noise excitation functions. In *Proceedings of the 1988 International Computer Music Conference*, Cologne, Germany, 313–21.

Rokeby, D. 2010. Very nervous system. Last accessed October 6, 2010, `http://homepage.mac.com/davidrokeby/softVNS.html`.

Schwarz, D. 2004. *Data-driven concatenative sound synthesis*. Ph. D. thesis, Universite Paris 6 - Pierre et Marie Curie, Paris, France.

Schwarz, D. 2006. Concatenative sound synthesis: The early years. *Journal of New Music Research* 35 (1): 3–22.

Schwarz, D. 2007. Corpus-based concatenative synthesis. *Signal Processing Magazine, IEEE* 24 (2): 92–104.

Schwarz, D. 2011. CataRT real-time corpus-based concatenative synthesis. Last accessed May 11, 2011, `http://catart.concatenative.net/`.

Schwarz, D., G. Beller, B. Verbrugghe, and S. Britton. 2006. Real-time corpus-based concatenative synthesis with CataRT. In *Proceedings of the 9th International Conference on Digital Audio Effects*, Montréal, QC, Canada, 279–82.

Schwarz, D., S. Britton, R. Cahen, and T. Goepfer. 2007. Musical applications of real-time corpus-based concatenative synthesis. In *Proc. of the International Computer Music Conference*, 47–50.

Schwarz, D., R. Cahen, and S. Britton. 2008. Principles and applications of interactive corpus-based concatenative synthesis. In *Journées d'Informatique Musicale*, Albi, France.

Singer, E. 2010. Cyclops. Last accessed October 6, 2010, `http://www.ericsinger.com/cyclopsmax.html`.

Sturm, B. L. 2006. Adaptive concatenative sound synthesis and its application to micromontage composition. *Computer Music Journal 30*: 46–66.

Torre, G., M. Fernström, B. O'Flynn, and P. Angove. 2007. Celeritas: wearable wireless system. In *Proceedings of the 7th International Conference on New Interfaces for Musical Expression*, New York, NY, USA, 205–8.

Vicon Motion Systems. 2002. *System Hardware Manual*. Unit 14, Minns Business Park, West Way, Oxford, United Kingdom: Vicon Motion Systems Ltd.

Vigliensoni, G. 2010. The enlightened hands: navigating through a bi-dimensional feature space using wide and open-air hand gestures. Montréal, QC, Canada. MUMT620 graduate seminar final report, Schulich School of Music, McGill University.

Vigliensoni, G., and M. Wanderley. 2010. Soundcatcher: explorations in audio-looping and time-freezing using an open-air gestural controller. In *Proceedings of the International Computer Music Conference*, New York, NY, USA, 100–3.

Von Hardenberg, C., and F. Bérard. 2001. Bare-hand human-computer interaction. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, 1–8.

Waisvisz, M. 1985. The hands, a set of remote MIDI-controllers. In *Proceedings of the 1985 International Computer Music Conference*, Vancouver, BC, Canada, 313–8.

Wanderley, M. M., and P. Depalle. 1999. Contrôle gestuel de la synthese sonore. In H. Vinet and F. Delalande (Eds.), *Interfaces homme-machine et création musicale*, 145–63. Paris: Hermes Science Publishing.

Winkler, T. 1997. Creating interactive dance with the Very Nervous System. In *Proceedings of the 1997 Connecticut College Symposium on Arts and Technology*.