Internship Report

Rithesh Kumar R

Advised by:

Dr.Richard Kronland Martinet, M.Thierry Voinier, M.Charles Gondre

Laboratory of Mechanics and Acoustics, CNRS, Marseille, France

Prof. Mohanaprasad K

VIT University, Vellore, India

12 Mar-7 July 2014

Acknowledgement

I would like to thank my parents for making this internship possible. I am nothing without them. My sincerest gratitude to my advisors Dr. Richard Kronland-Martinet for being an inspiration, M. Thierry Voinier for the immense support and encouragement and M.Charles Gondre for being patient and kind. I owe my work to the entire LMA team for holding me. My deep respects and gratitude to my guide Prof.Mohanaprasad K for being a mentor to me. I would like to thank M.Lennie Gandemer for being the friend that he is, my stay in Marseille was great because of his kindness. My deep wishes and thanks to my fellow friends/interns M. Tristan Begueria, M.Clément Ghirardi and M.Thomas Bordonné. Last but not the least, I dedicate this to my best friends Ms.Dheepa, M. Shahul Hameed M and M.Ashwin Kumar N

The Singing Baguette

Contents

Introduction

Chapter 1

Introduction

1.1 Initial Instruments

1.1.1 The Beginning

We are in en era of Computer Music Controllers, reaching multiple possibilities, far beyond the capabilities of conventional music instruments. The idea arose from using electricity to refine or change the tone of an existing instrument. The design of the first ever known electronic music instrument dates back to the 18th century, when Denis d'or invented a 700-string keyboard that electrified the strings temporarily to improve the sonic quality in 1753. A hundred years later, Elisha Gray invented the Musical Telegraph in 1876, which was nothing but an electric telegraph for transmitting musical notes. This is the first root for the modern day synthesizers, for their design and ideology. That is why Elisha Gray is known as the father of the modern day synthesizer.

In 1926, the Soviet Scientist Léon Theremin invented a ground breaking instrument: the Theremin. It was the first ever instrument that worked on gesture recognition. It stands out by the fact that it is played without any contact. It has two antennas and the proximity of the musician's hands from each of the antennas determine the pitch and the volume of the sound respectively. Needless to elucidate, the antennas were nothing but proximity sensors. Another important instrument was the Licht-ton Orgel. This invention from 1936 was a digital organ that played samples from analog optical discs. From then on, there have been multiple innovations and inventions in the field of computer based music instruments.

The Hammond synthesizer manufactured by the Hammond Organ company 1930s, offered polyphony and had 12 oscillators with a basic envelope control along with serially program-

Figure 1.1: Musical Telegraph: Prototype of the modern era

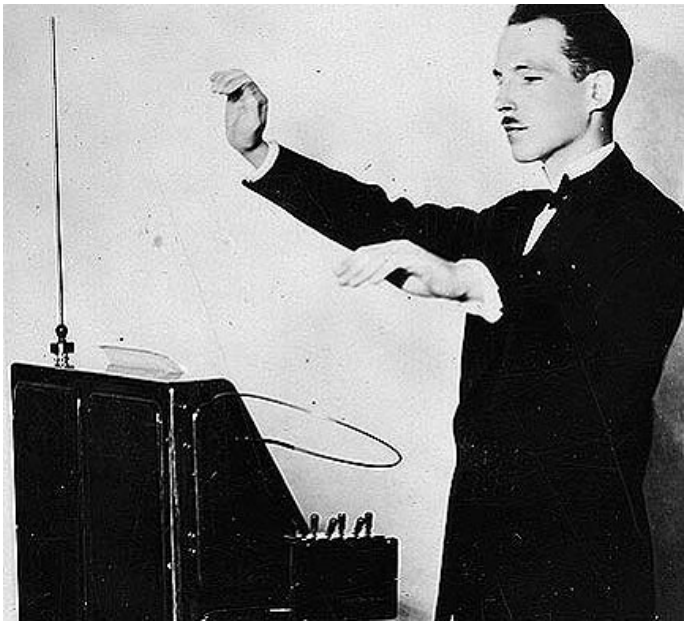

Figure 1.2: Gesture Recognizing Music Instrument

mable low pass filters. It was based on the Telharmonium, invented 30 years ago by Thaddeus Cahill. They are based on electromechanical tone wheels. They are an assembly of motor driven wheels induced with electricity along with a sound pickup. The rotation of the geared tone wheels, along with the induction of electricity produces controllable sounds.

The 1960s and the 1970s were the period of Analog Sound Synthesizers. Synthesizers became a crucial part of the music industry, owing to their capability to produce myriads of sounds just by tweaking the necessary parameters. However, the start of synthesizers was from samplers. Samplers are instruments that play recorded samples based on the trigger. One such notable sampler was the Mellotron. It played the audio from recorded tapes, so when a key is pressed, the relevant sample would play; much like a tape recorder. The mellotron had the possibility to adjust the pitch and tone of a sound and to mix several recorded tracks: this is where the possibility of sound synthesis arose. An updated version even facilitated removable tape frames, and thus one had the flexibility to place any sound that they wanted.

1.1.2 Computer Musical Instruments

Although such analog synthesizers were used both in studios as well as in live performances, they were very expensive and heavy. The advancement in computer technology led to the birth of digital synthesizers. One of the most successful and the first polyphonic digital synthesizing engine was the Fairlight CMI. The term CMI stands for Computer Musical Instrument and thus the birth of the generic term. It had a visual representation of the waveforms being synthesized and ran on its own operating system, facilitating a menu driven GUI. The birth of CMIs lead to dedicated sound processors and software synthesizers.

By the 1980's computers were not only used to make sounds, but to also compose a music soundtrack completely. This was the time that modern day music signal processing took its form. Miller Puckette in the mid 80s developed the Graphic based Music Processing Software "Max", at IRCAM, Paris. This entire project has been done almost entirely using max. Digital Signal Processors, commercially known as sound cards came into existence. Sound cards are supplementary processors to a computer that are completely dedicated to processing sounds/waveforms. The famous MIDI, the standard protocol for trans platform music instrument note representation and communication, was also invented in the same decade. MIDI stands for Musical Instrument Digital Interface. The advent of MIDI and software based instruments, made sophisticated technology affordable.

Figure 1.3: Fairlight CMI:The Visualisation Breakthrough

1.1.3 Modern Age CMI

The recent times have taken innovation to the next level; apart from sound synthesis other considerations such as physical controllability, portability, aesthetic representation came into focus. The synthesizers such as the Bodysynth are based on physiological computing, producing interactive art. Another instrument, called the CubeLife is an interactive art instrument that is entirely based on the participant's heartbeat [1]. It is a cube that creates an artwork depending on the heartbeat, using a sensor along with its own synthesis algorithm. The artwork can be a sonic/visual piece or both.

Another notable instrument is the Nukulele. It models an existing instrument, namely the Ukulele, but also extends it to a digital realm increasing its range and sonic controllability. It uses a Karplus-Strong plucked string model, controlled by linear force sensors, to simulate the interactions of a virtual string [2]. There were also even more intuitive and simple inventions such as the PhiSEM controllers on a Frog Maraca Frame. It was otherwise known as the haptic maraca. The Physically-Informed Stochastic Event synthesis Model (PhiSEM) is an interconnection of accelerometers, force sensing resistors and switches that are meant to translate the shaking and scraping gestures into sound.

1.1.4 Designing the instrument

The history and the development of computer musical instruments suggested us various things to keep in mind while designing our own instrument. Each of the case studies done here, teach us various things to learn from. The fairlight CMI, although being a critical success, did not make it commercially. In fact, the company went broke in a few years. The reason cited was the cost of manufacture that was involved. And yet, its success was mainly credited to its visual appeal: for its ability to depict the waveform and to visually see the modifications that are done using the synthesizer. So we do understand that our instrument should be easy to manufacture and have a Graphical User Interface.

Another case is of the theremin. The instrument's phenomenal achievement is its contactless playing and portability: it is so successful that it has ever found its place in a classical symphony orchestra. Our learning is that making a gesture recognizing CMI is not only interesting to the user, but is also a lot easier to work with and manipulate for the easier. It is also important to make a fairly portable system. The modern day Haptic Maraca and the Nukulele suggest that using everyday objects is far more useful than to build a new design. It not only makes it practical for the user, but also to the manufacturer/developer, relieving oneself of the complications involved in making a novel prototype. Further, it can also act as a dual purpose instrument.

Figure 1.4: Modern CMI: PhiSEM Maracas

Although we have multiple affordable processors in the market today, soundcards are still uncommon. So, the focus is to make our process minimal on the chip. That is, we wish our

work to function on the most mundane of system requirements, making it available to all. This project work never uses an external soundcard, despite its functionality of making myriads of sounds. This ability is mainly attributed to the engine in which the project is run on, namely, the software Max 6, which is a fast and yet a light software.

In a broader context, what we aim is to design a new musical instrument, in its own regard. Looking into the challenges faced in creating a new instrument, the following are the primal constraints on the design:

1. To co-design with an existing instrument, thereby extrapolating its possibilities
2. To include all the parameters but to also not have few-to-many mappings
3. Inculcating everyday objects, like the PhiSEM controllers that have the ‘Frog Maraca’ frame.[3]
4. To have a dynamic sound gesture relationship

Typically a digital musical instrument model is characterized as [4]:

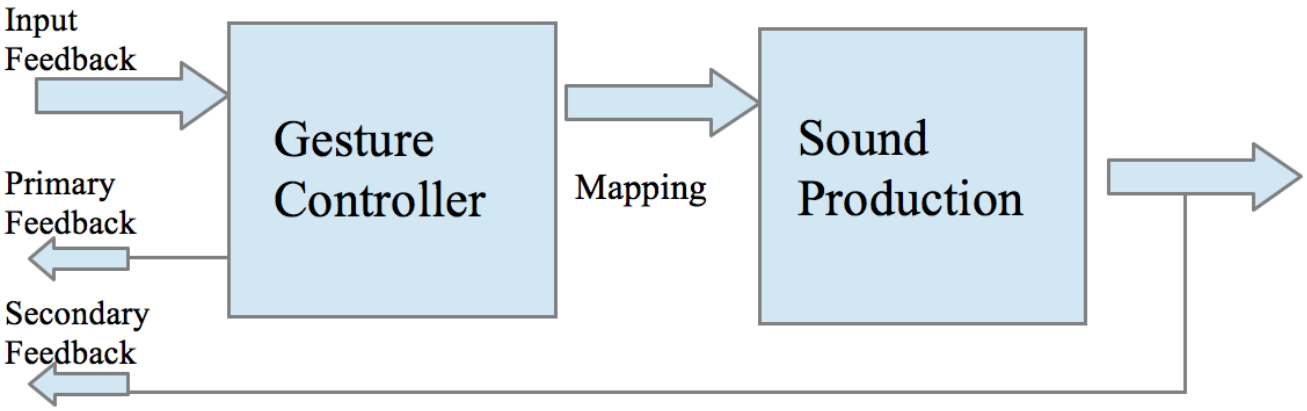

Figure 1.5: Characteristic of a digital music instrument

The fundamental technical issue is the conversion of analog/digital electronic signals into control data useful for computerized sound generator [5]. We are making use of the three coordinate three parameter continuous input from the sensor device, embedded into a plastic drumstick. On the other hand, the digital synthesizer must have adequate possibilities to meet our design expectations. The indigenously developed 'Tapong' synthesizer is an intuitive system that inculcates the action (rolling/rubbing/scratching) and the material involved, to name a few of parameters. This makes it easy to relate to the gestures and the subsequent synthesized sound.

In traditional instruments, the relationship between gesture and sound is one-to-one: a single action triggers a single sound; however, in computer mediated instruments, a single trigger can have any result [5]. We neither aim to complicate the actions, nor do we want to stick to the conventionality. A trigger must be powerful enough to produce flexible sounds but yet maintain the simplicity.

MAX/MSP: The Development Engine

Chapter 2

MAX/MSP: The Development Engine

2.1 What is Max?

Max/MSP is a visual programming language that helps us to build complex, interactive programs even if one does not have prior experience in writing coding. Max/MSP is especially useful for building audio, MIDI, video, and graphics applications where user interaction is needed. Max is the environment that is used to create visual programs, called patches, plus a set of building blocks (called objects) used in those programs. MSP is a set of Max objects for audio and signal processing. Jitter is a set of Max objects for video, graphics, and matrix data processing.

2.2 Why Max?

Max has the reputation of "It can do anything you want to", when it comes to manipulating and handling sound. But it is also true that many of the things that one could do in Max are much more easily and efficiently achieved with other more polished pieces of software. But there are the unique abilities that Max has that other programs do not offer. It is a lighter process in comparison with other such audio related programs. In my personal opinion and experience, although MATLAB is a powerful tool, Max handles audio processing much more easily.

When it comes to the relevancy to our work, it stands out in its ability to handle input data. This ability, to handle live incoming data and have it affect output, gives a platform from which to create performance patches that are adjustable in real time, interactive performance

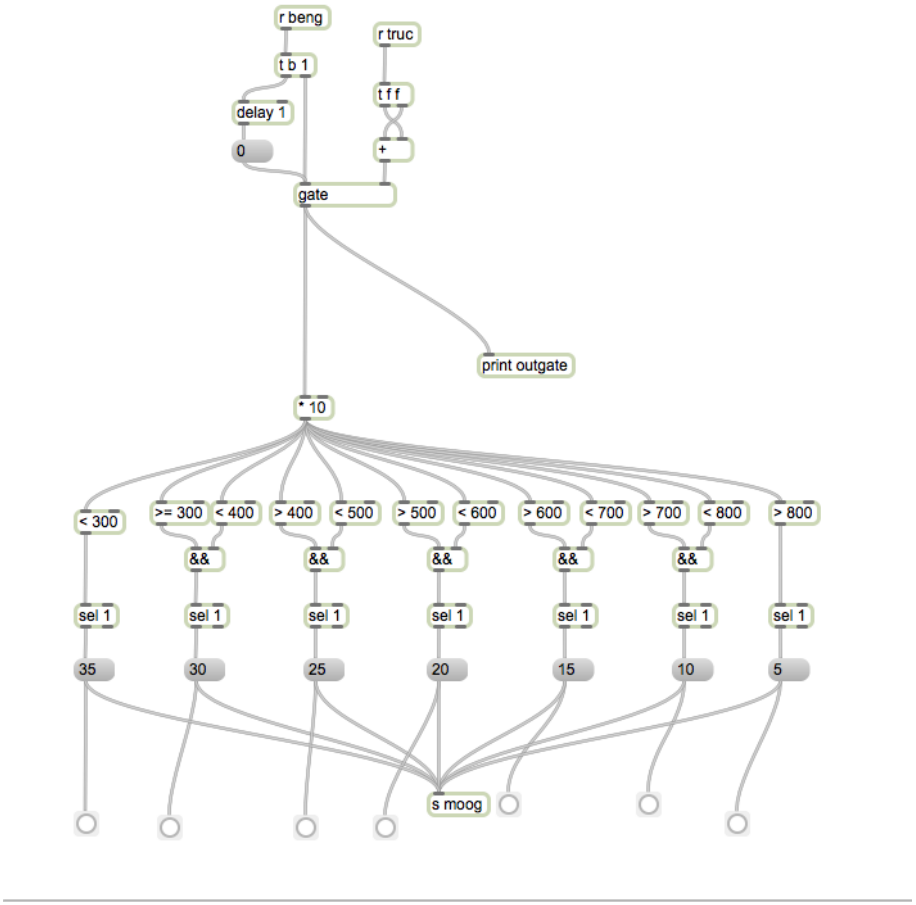

Figure 2.1: A simple max patch that was used in this work

situations for musicians, interactive installations, cross continental web based performance patches, and the list goes on and on. The software provides myriads of possibilities of interoperating between various programs(patches). It is a dynamically operating programming language and invalidates the need to compile the program every time; the changes are reflected as and when they are made. We aim to have a live system, where data input and feedbacks keep happening simultaneously. The work is in fact based on the continuous interaction between the user and the computer. Max is exactly the right platform that suits our needs.

Tapong: The Synthesis Engine

Chapter 3

Tapong: The synthesis Engine

3.1 What is Tapong?

The Tapong is a synthesiser that has been developed by our team at Laboratory of Mechanics and Acoustics. It has been made for musical and virtual reality purposes, offering an intuitive control over generated sounds. Since the Tapong has been entirely built using the Max/Msp engine, it facilitates the possibility to manipulate the intrinsic characteristics of sound in real time by controlling a few parameters. It is essentially an additive synthesiser that functions using a three layer control strategy.

3.1.1 Three layer strategy

The top layer of the synth is the verbal description of the kind of sound imagined. It gives an intuitive way for the end user to create sounds by specifying the perceived properties of the object involved, along with the kind of impacting action. The next level involves the descriptors related to the characteristics of the signal. That is you could specify the parameters for generalized function of filters, rather than directly operating on them. The parameters specified roughly translate according to the verbal description given by the user at the top layer. The last layer, consists of the interconnection of the resonant filters, wherein the user can directly specify the basic parameters to each of the filters.

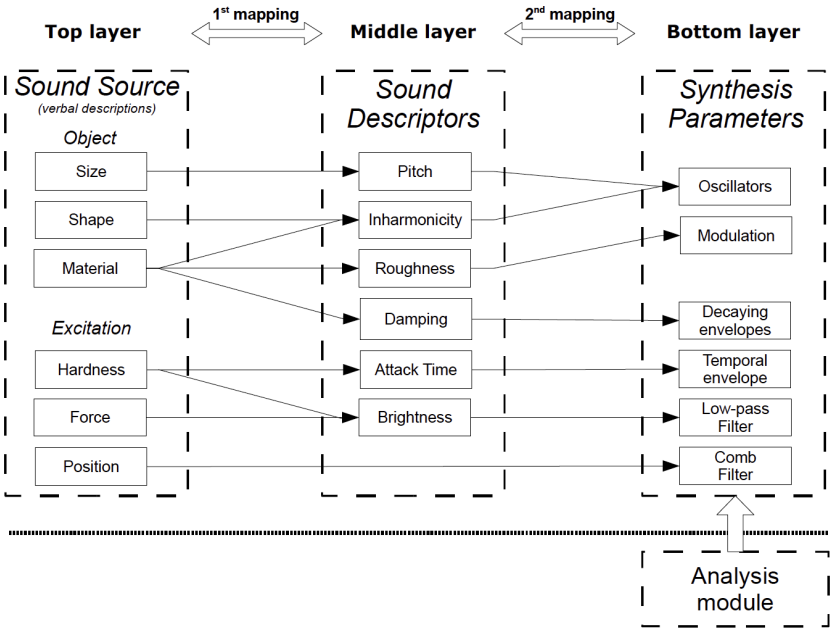

Figure 3.1: Flow diagram explaining the three layers of Tapong

3.2 Synthesis Algorithm

The synthesizer is based on the idea that the sounds made by interactions with a surface is a series of impulse sounds, that are generated when an object (the finger) is run in contact with the material [6]. Therefore the generic sound synthesis model is describes as a summation of time varying impact events, with specific relations:

$$f(t) = \sum_{n} A^n \phi^n (t - T^n) \tag{3.1}$$

where A^n and T^n are respectively the amplitude and time-position of each impact, and ϕ^n represents the "impact pattern", i.e. the shape of the n^{th} impact. The following diagram elucidates the additive synthesis pattern:

The input signal consists of two contributions: Tonal and Stochastic. The tonal contribution is provided by combining a set of 96 sinusoid oscillators and 96 narrow band filtered noises. The respective outputs of the sinusoids and filtered noises can be mixed craftily using a fader (precise/blur control). A noise generator provides the stochastic contribution. The noise parameters can be specified by the user and is usually a Gaussian noise. The output levels of stochastic and tonal contributions may also be adjusted by a fader (tonal/noisy control). This mixture is passed through a series of resonant filters, providing exponentially decaying envelopes. There are a total of 24 envelopes, one per bark band. Bark bands are psychoacoustical

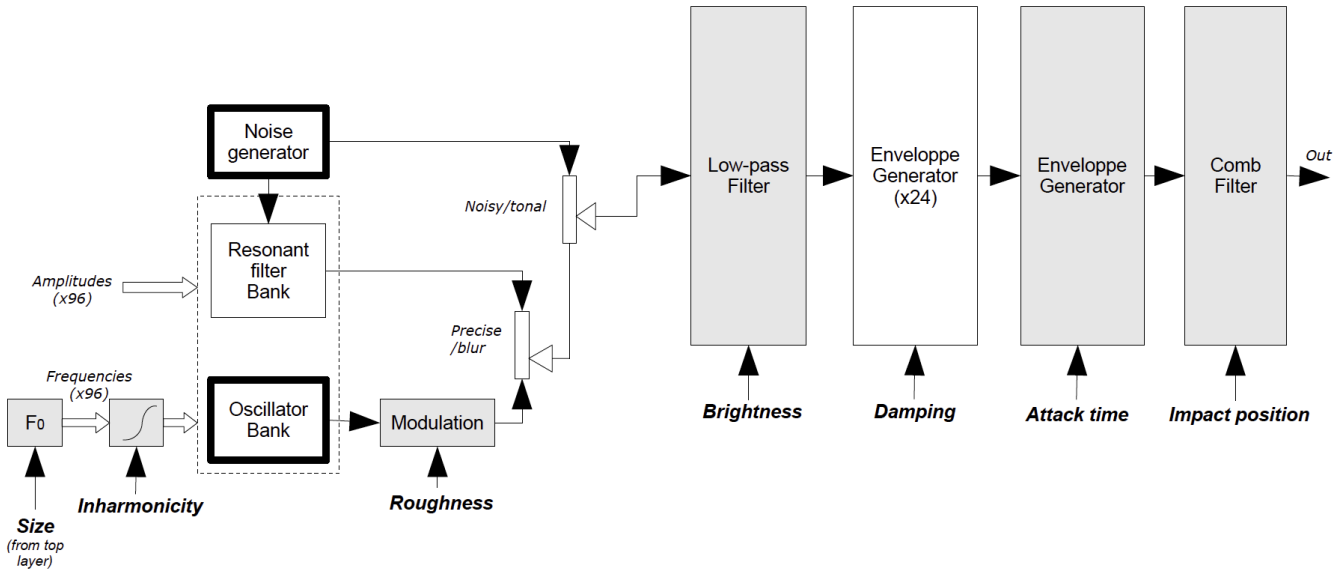

Figure 3.2: Sound Synthesis Flow

bands, spaced around the centre frequencies of musical notes. The same envelope is applied to the stochastic and deterministic parts of these bands, so as to increase the merging between them. The final signal is passed out using the requisite post process filtering. In order, to have a purely deterministic signal, the envelopes for the stochastic and deterministic parts can be applied differently, but is omitted at this level, to avoid complication.

However, this approach is now obsolete. This is because, of the complexity and lesser controllability of the additive synthesis. If one wanted to create an impact sound, then we have to go through multiple envelopes to filter out an impulse from a continuous input from the oscillators. It becomes harder to make any subtle changes, due to the sophistication in the process. Therefore, we shift to subtractive synthesis, and give an input identical to desired output envelope. We directly feed noise/impulses to the system to get a continuous/impulse output respectively. The synthesiser works on a new flow, as depicted in the diagram, in Figure 3.3

In summary, the synthesiser is an amalgamation of a group of parallel connected resonant filters [7]. The natural response of the filter is a decaying sinusoid, with an exponential decay. Since the decay changes with frequency, the decaying factor is adjusted to facilitate a single response [6]. Based on the parameters mentioned by the user, a mix of input is sent to the filter bank. For example, if the action is set to impact, an impulse is sent to the bank. If it

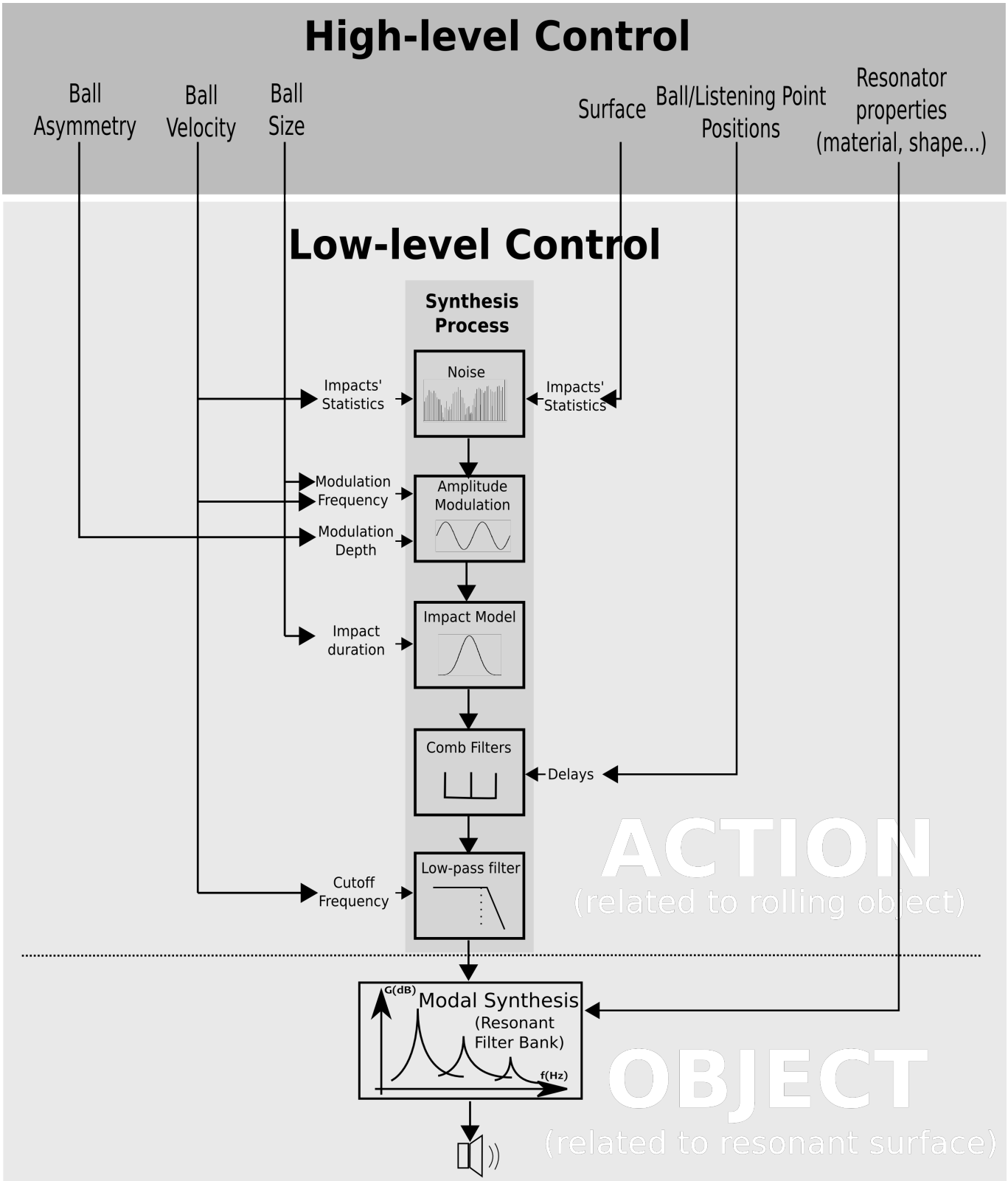

Figure 3.3: Sound Synthesis Flow

is scratching/rubbing/rolling, a series of impulses [6] is sent.

The resonant filter used is a team developed max patch, that produces decayed sinusoids at a given centre frequency and damping factor. It also inculcates a gain factor, to adjust the amplitude of the output. The filter is a two pole filter, that has been designed to eradicate the discontinuity problems during the transience [8].

3.2.1 s2m.resFS1~

The beauty of the synthesizer lies on the fact that it is completely original from the bottom level up. It is a second order high Q digital resonator, based on [8]. The reason that a high Quality Factor Filter was chosen is because, most musical instruments involve high Q mechanical filters [8]. The filter is based on the transfer function:

$$H(z) = \frac{R \cdot sin(\theta) \cdot z^{-2}}{1 - 2R \cdot cos(\theta) \cdot z^{-1} + R^2 \cdot z^{-2}} \tag{3.2}$$

where $\theta = 2\pi f$ is the frequency, R = $e^{-\alpha T_e}$ relates to damping (alpha – damping factor, Te is the time period of the sinusoid) Z is the input signal in the complex number representation. Since the representation is in the frequency domain, the gain can be easily achieved by simple multiplication. So, the resonant filter has 4 controllable parameters.

The resonant filter basically produces decaying sinusoids. Any type of signal from impulses to noise signals can be used to excite the filter, thus giving us multiple possibilities. The control of damping and frequency, give us a superior control over the signal characteristic. The most basic and yet vital aspect, the Gain level, helps us to mix the sounds from each of the filter at the right level, to make the resultant composite sound.

3.3 The interface

The intuitiveness of Tapong lies in its ability to model real life sounds. It emphasises on the Action-Object paradigm [6]. It bases on the concept that the sounds are resultants of an action on a resonant object. For instance, either plucking the string or bowing it can differentiate the sound from a violin. Here, although the material is the same, the action differs. The psychological theory underlying this paradigm is called the ecological approach of perception introduced by Gibson in 1966.

The interface now becomes self explanatory. The action module specifies the way the striker comes in contact with the material: either by rubbing/scratching/rolling/impact. The

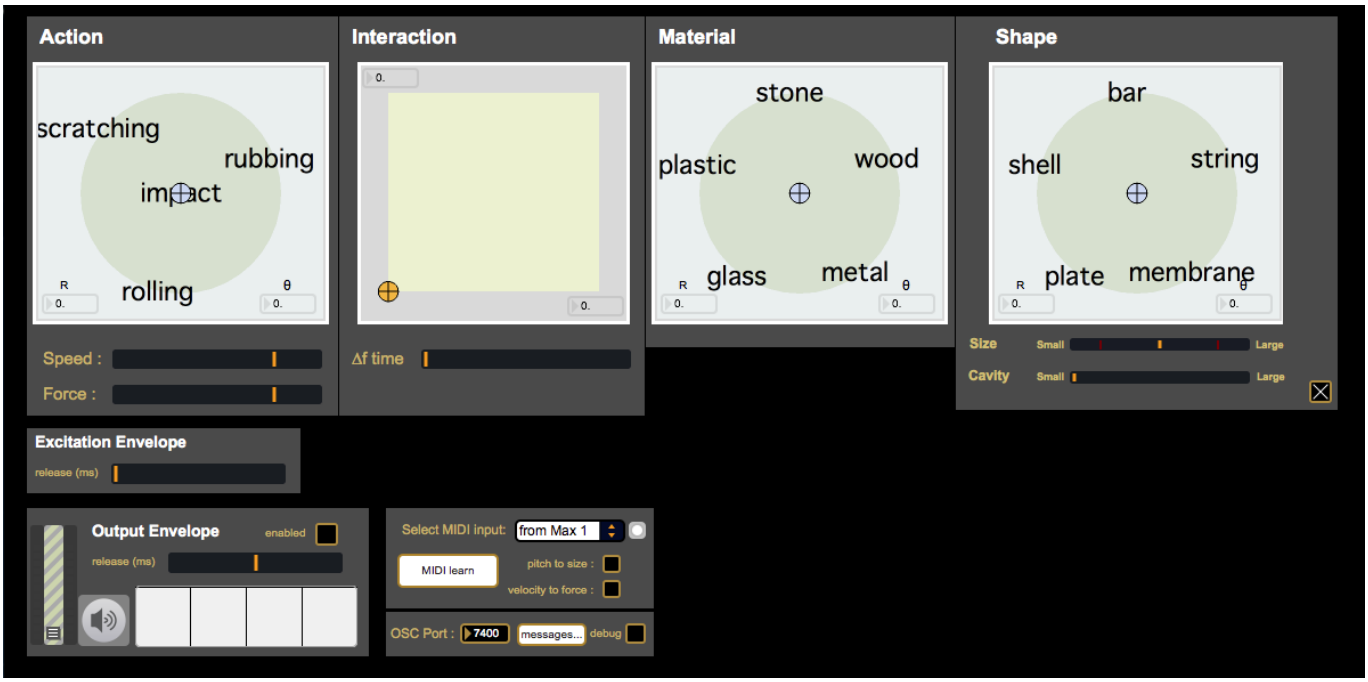

Figure 3.4: Tapong's Interface

interaction space defines the mix of tonal and stochastic contributions discussed earlier. Material: whether the object is made up of wood/glass/metal/stone/plastic. Shape: the shape of the object, as this is very crucial, as different shapes produce different reverb spaces and resonating sounds. Release is the parameter that specifies the time up to which the sound plays after the initial trigger. The release of the output and excitation (input) envelopes can be controlled. The synthesiser also supports MIDI. This means that, any external computer music instrument could be used to control the synthesiser parameters.

Baguette: The gesture sensor

Chapter 4

Baguette: The gesture sensor

4.1 What is the Baguette?

The gesture sensor is an Inertial Measurement Unit (IMU) consisting of four sensors. The gesture sensor, with a small dimension of 50 x 10 x 4.56 mm, has been embedded into a drumstick, to extend the possibility of an existing instrument [3] and also for its dynamic possibilities. The sensor used is an MPU-9150 commercially made chip, manufactured by InvenSense. This chip has been embedded into a board along with a micro controller, which is in turn connected to a Bluetooth module. It has been carefully designed to fit into the base of a drumstick. The board is custom made. The chip constitutes of: the Gyrometer, Accelerometer and the Magnetometer; each yielding a three dimensional output [9]. It also includes an additional temperature sensor, for convenience and in case there has to be any temperature compensation [10] of the gyrometer. The data is continuously sent with a precision of 4 decimal points, via a serial port using bluetooth at a baud rate of 115 Khz. The chip runs on a supply voltage of -.5 to 6 V and the battery used provides a substantial life of about 5 Hours per charge.

Figure 4.1: The IMU and the drumstick

4.2 Working with the Baguette

The data received is interpreted by a maxpatch, whose interface is shown on figure 4.2.

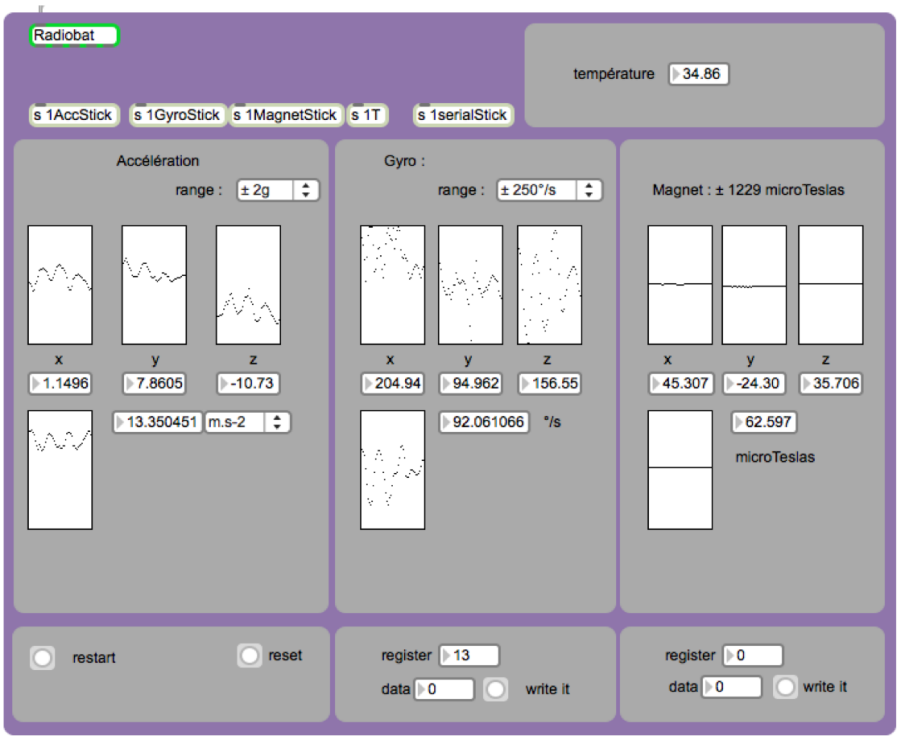

Figure 4.2: The Interface of the Baguette

The interface offers the flexibility to choose the data to process from. It also can pass commands to the sensor to modify the range and to set the mode of running. The modes include reading and writing of the gyro/accelero or magneto registers, to restart the chip and as well as to receive the whole data stream. The data stream received is of 24 bytes long and is the data from three sensors come as 6 bytes for each: each coordinate consists of an LSB and MSB, thus yielding 6 bytes for the three coordinates. The data is sampled at a frequency rate of 100 Hz and takes 10 ms to reach the computer from the chip, given the baud rate, thus creating a latency. The frame is depicted as follows:

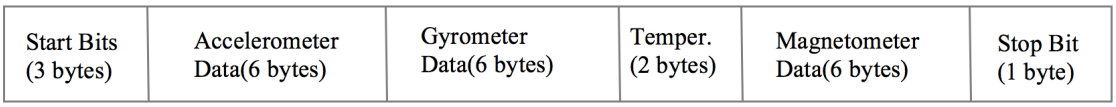

Figure 4.3: The Chips's Sensor Frame

This received stream is decoded and represented in the interface. The dataset is the tri-axial Cartesian coordinate system and the received data range can be adjusted. The maximum ranges are as follows: ±16g, ±2000°/s, ±1229 μTeslas. The interface provides a visual representation, and it was a major source for the data study. We can also tap the absolute magnitude of the coordinate values from the interface, if necessary.

Building the new instrument

Chapter 5

Building the new instrument

Now that we have analysed the tools in hand, the mission now is to link the gestures with the synthesiser. There are two broad approaches on the gestures involved: firstly, the impacts; next are the actual gestures of movement. The focus is on developing foolproof, robust gestures, rather than creating multiple gestures with degrees of uncertainty. The simplest and consequently, the most intuitive of them all is impact recognition. The drumstick must recognise when being struck and must produce an amplitude of sound analogous to the striking force.

5.1 Data Analysis

Before actually working on recognising the gestures, we must understand the flow of the input stream received. This fact was however understood after a considerable time spent on research, working directly on recognition. As mentioned earlier, we receive the input of the three sensors in the Cartesian coordinate system. The additional temperature sensor renders useless in this situation, as ambient sensors are subject to change. Also, tests showed that there really wasn't any significant change in the temperature, even during fierce movements of the stick.

Therefore, calibration of the stick was important. A week's study on the received data with respect to various movements revealed greater insights into the task. Initially, we defined various simple gestures based on which further gestures could be built. There were four gestures in mind: 1) Strike on air/object; 2) Circle; 3) Wave; 4) Roll. On doing these gestures, the inputs from each of these sensors were analysed and the possible patterns to characterise

each of these gestures were studied.

The magnetometer was not really helpful, as it did not sense any significant information in our study. The analysis was thus narrowed down to accelerometer and gyroscope. The accelerometer made sensible input: when there was an impact, the acceleration of the stick soared and changed directions suddenly, causing jerks. This can be easily explained, as the stick after the impact rebounds and thereby undergoes acceleration in the opposite direction. So whenever there were sudden direction changes, there were jerks in acceleration data. However, for the same kind of impact, there were jerks; but in a different axis every time, since the orientation of the stick is not stationary. Therefore, using the absolute value of all three axes was found suitable. But the paradox here is that, now knowing the direction of acceleration is not possible since we are calculating the absolute values.

When doing the other three gestures, acceleration more or less was an ideal sinusoidal wave input. So, if we were to device a method to recognise the input being a wave, recognition would be possible. The other sensor, the gyroscope, was mostly helpful in the rolling action. In other cases there were just stochastic input. When you roll the stick by tossing it into air, there is enormous angular displacement. The orientation alternates as it rolls, and therefore there is a switch between the maximum and minimum values of the sensor. This more or less resembled a square wave. So, with this understanding, we proceeded to work on linking the synthesiser with the sensor system.

5.2 Recognising Impacts

When an object is struck, its acceleration changes with respect to time. Although its angular momentum changes, it is also dependent on the axis of rotation of the object itself. So using the accelerometer data to analyse the impact is more relevant. Several approaches were used to interpret the data:

1. Use just the acceleration along the x axis: the data, although it was in concurrence with the direction of motion, it did not give adequate results. The draw back was that the level of stationary position could change, depending on the angle of rotation

2. Use the absolute value of the three axes: this was very efficient input, however, the bottleneck was that, since it is a modulus operation, even a deceleration would produce an output identical to accelerated impact.

Clearly, just using the direct input to process for recognition, was not reliable. Therefore, the input must be manipulated, so that we only get the right data to be used for recognition. Several methods were tried:

1. The differential of the absolute value.
2. The modulus value, only when all the three axes were positive signed.
3. The differential of each axis, and then all summed together.

Although these methods gave us a refined input, they still failed to generate a robust recognition. At all cases, it must be understood that the recognised impacts were sent to the synthesizer to play impact sounds.

So, it was then understood that the values must be normalized, so that despite the change in orientation, the stick behaves as though it yields output from an ideal position. This will eradicate the above ambiguities, and enunciate the direct interpretation of the data. Since we are focusing on the upward and sideward movements of the stick, its angle of rotation is irrelevant. Therefore, using the angles of azimuth and elevation would render more information, rather than the Cartesian coordinates. The data was converted to polar coordinate system for analysis, to research if it made more sense to the impact recognition.

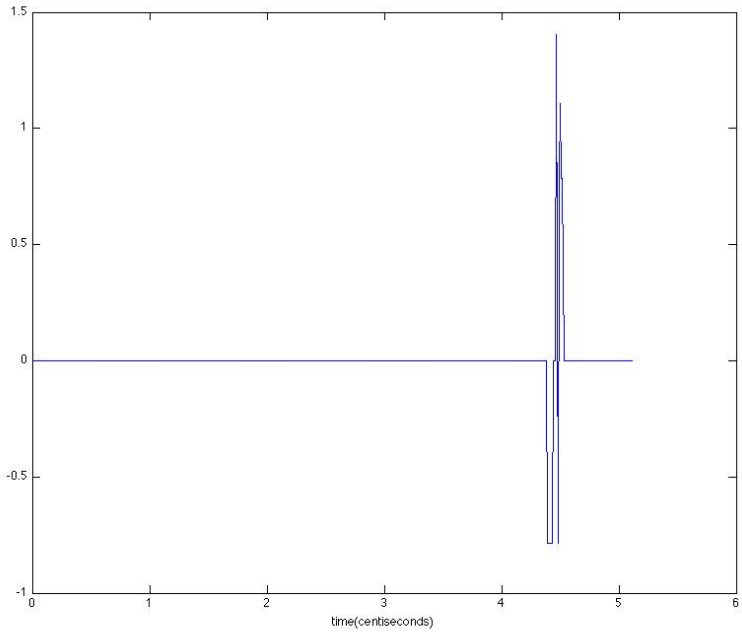

Figure 5.1: Impact representation in Polar Coordinates

It was understood that the polar angle had a zero crossing of double edges on either sides, for one impact, every time the stick struck an object. This however, was not the same case for an impact on air. Using this data, one can find the event of an impact, and also differentiate between whether the strike is on a solid object or on air. Subsequently, it was necessary to find out the magnitude of impact. And this did seem easy, as our parameter disregards the orientation of the sensor but only parameterizes the position with respect to the horizontal reference, which is parallel to the ground surface.

So, new methods were sought to find the force of impact. It was quickly realised that the differential of the elevation angle is not sufficient to measure the force. Although the absolute resultant of the accelerometer is helpful in measuring the force, it however has an ambiguity: we will not be able to tell about the direction of the force. So, a force in any direction with the adequate amplitude, can be mistaken for an impact.

Even if we are able to tell the direction of force, the direction is always with reference to the horizontal of the sensor, and not necessarily to the ground surface. Therefore, it was proposed that the gyro-metric data be used to correct the position to the ideal state.

However, the gyro data is angular velocity, and therefore only shows the magnitude of rotation and not necessarily the relative position of the stick. Only the accelerometer data gives an idea about the absolute position of the stick. Although the accelerometer data is theoretically feasible, it did not yield the expected output. The reason however was not ascertained. At this moment, the impact recognition was robust, but however, the force detection was not as expected. This was so because, occasionally for a small impact, it gives the result for a big impact, and vice versa. There was confusion as to whether it was a problem with the stick or if the data interpretation had to be changed.

With repeated experimentation and prolonged trials, it was understood that the sampling was not sufficient enough to characterise the minute changes; Increasing the data rate was also not possible, as it was the maximum data rate of the Bluetooth connection. This is when we inculcated a firmware upgrade; it is described as follows:

1. The chip functions at 1kHz, but the Bluetooth transfer rate and thus the sampling rate is retained to 100 HZ

2. Therefore, the chip has 10 values, every 10 ms. So now, the mean of these values is sent via the serial port

3. Also, the modulus derivative of acceleration is calculated by the microcontroller and the max of the ten values is sent. This is to find out the sudden drifts in acceleration, known as Jerks, that occur during impacts/sensor vibrations

In short, the chip doubled as a vibration sensor. So, rather than working on the data from the sensor, the sensor works on that data by itself. It works on the concept of jerks. Jerks are sudden drifts in acceleration, which are caused in cases like the impact. When the chip calculates the derivative, it in turn calculates the magnitude of the jerks involved. An important point to note is that this does not depend on the movement that causes the impact, but rather on the vibration that is involved at the time of impact. Hence, our recognition has narrowed its realm and thus making it better.

The force of impact also comes as a byproduct of this process, as the jerk calculated by the chip translates to the magnitude of impact. This is an interesting outcome, as we have made futile attempts in the past to calculate the force of impact. In the past, there was lots of false recognition, due to the absolute value error of acceleration, discussed earlier. With this current outcome, however, there were latency errors, which are usually expected. This is because, the sound generated by the impact and the force associated with the impact were getting mixed up. The source of this error was not discovered because the focus lied on robust impact recognition and not on the force of impact.

The amendment in the chip had rendered useful and could now be used easily to detect impact on an object and discard the one in the air. This, along with the acceleration data was helpful in setting the threshold for recognising the impact. The recognition was absolute when struck on to an object; but however, there were false detections when there were erratic movements in the air. Nevertheless, this system proved to be effective in general, and there was no false impact recognition when performing the proposed gestures.

5.2.1 Double Impacts

The accelero values were manipulated to find if two successive impacts were caused using the stick. This was done by setting a threshold on the acceleration value. The incoming values were summed up and compared with the threshold. Since we have jerks when making impacts, it in turn means a higher summation. The threshold was fixed by trial and error. An arbitrary time period was fixed, within which the two impacts should happen. The summation was reset to zero at the start of every period. Once the impact was realised, the tapong was

sent messages so as to change the material. The advantage of tapong is that, we could send messages to the synthesiser and control it, thereby facilitating external control.

5.3 Recognising Gestures

It suffices to recall the gestures that are to be recognized, which are the focus of this section: 1) Circle; 2) Wave; 3) Roll. As mentioned in the subsection, data study, the close study of accelerometer data shows that a circular motion yields wave-like data stream. So, in recognition of gestures, the foremost study was to characterize the data stream: to algorithmically realize that the input stream is a sinusoid. Intuition tells us to analyze the frequency spectrum, in other words, to perform Fast Fourier Transform on the input. A considerable time and energy was spent on this process, helping us understand that it was not practically possible using the tools that we had. The bottleneck was that our data rate was not high enough. One had to wait for 2.5 seconds to even start the process (256 samples for a minimal FFT resolution, given that our data rate is 100 Hz). There were implementation difficulties as well, as MAX views the real-time data set as a series of integers, and not as a valid signal (mathematical function).

It also did not take long enough to realize that trying to recognize gestures with the raw data set was not a feasible option. Another representation was definitely necessary. The sensor had always been baseless. Although we knew what was being done to the sensor, we did not know how it moved. There was no idea about its orientation. This curiosity led to discovering and implementing the Orientation filters [11] that usually find their use in aircrafts.

5.3.1 The new algorithm- Attitude and Head Referencing System

The sensor system can be referred to as a Magnetic, Angular Rate and Gravity (MARG) system; it is essentially an extension of the Inertial Measurement Unit (IMU); the extension being a magnetometer. It provides a complete measurement of orientation of the stick, with reference to the direction of gravity and earth's magnetic field [11]. So we are using an orientation filter that estimates the position of the stick, fusing the data from the three sensor measurements. Unlike the regular Kalman-Filter based systems, this filter uses a Quarternion representation to compute the estimation.

5.3.1.1 Quaternions

Quarternions are a convenient way to represent the orientation of an axial system, with respect to a reference axial system.

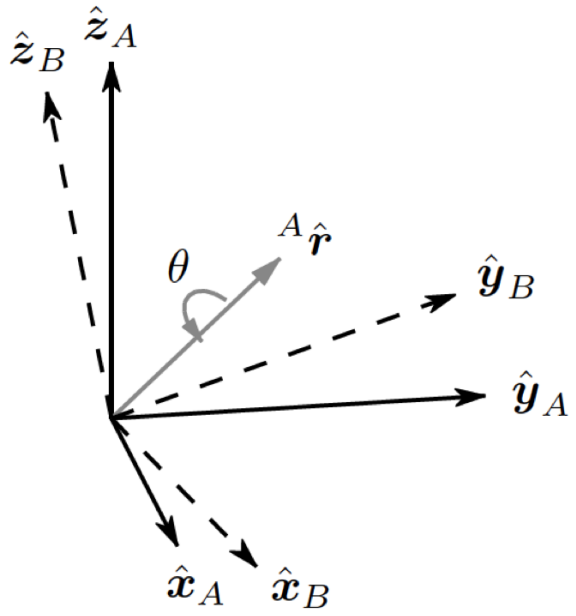

Figure 5.2: Figure explaining the concept of Quaternion

A quarternion is defined as:

$$ {}^A_B\hat{q} = [q_1 \quad q_2 \quad q_3 \quad q_4] = \left[cos\frac{\theta}{2} \quad -r_x sin\frac{\theta}{2} \quad -r_y sin\frac{\theta}{2} \quad -r_z sin\frac{\theta}{2}\right] \tag{5.1} $$

where the symbol ${}^A_B\hat{q}$ represents the orientation of frame B with respect to frame A. The first scalar q_1 represents the angle of rotation about the axis of unit vector ${}_Ar$ and the remaining are its coordinates [12].

The converse, which is, the orientation of A with respect to B is given by the conjugate:

$$ {}^A_B\hat{q}^* = {}^B_A\hat{q} = [q_1 \quad -q_2 \quad -q_3 \quad -q_4] \tag{5.2} $$

5.3.1.2 Algorithm Implementation

The first task is to find the reference axis (earth frame). The gravity and the direction of the earth's magnetic field can be found out easily using the accelerometer and gyroscope respectively. Subsequently, this is utilized to get the reference axis.

However, for any given measurement in the sensor, there would be infinite solutions, along the actual reference axis. This situation is counter acted by using a presumed vector, in the earth frame.

The magnetic field in the earth frame is represented as that of having a vertical (z-axis) and horizontal components and the gravity as that of the vertical component alone.

Our aim is to find the quarternion, ${}^{S}_{E}\hat{q}$ that represents the orientation of the earth frame with respect to the sensor frame. There are two possible ways to achieve this:

(i) We begin calculations from a presumed direction of field (both magnetic and gravitation) in earth frame ${}^{E}\hat{\mathbf{d}}$, and also an initial quarternion using the measured direction of field from the sensor ${}^{S}\hat{\mathbf{s}}$, and hence iterate to estimate the value. The desired quarternion is defined as the solution to the function:

$$\min_{{}^{S}_{E}\hat{\mathbf{q}}\in\Re^4} f({}^{S}_{E}\hat{\mathbf{q}}, {}^{E}\hat{\mathbf{d}}, {}^{S}\hat{\mathbf{s}}) \tag{5.3}$$

$$f({}^{S}_{E}\hat{\mathbf{q}}, {}^{E}\hat{\mathbf{d}}, {}^{S}\hat{\mathbf{s}}) \;=\; {}^{E}_{S}\hat{\mathbf{q}}^{*} \otimes {}^{E}\hat{\mathbf{d}} \otimes {}^{E}_{S}\hat{\mathbf{q}} \;-\; {}^{S}\hat{\mathbf{s}} \tag{5.4}$$

This function is optimized using the gradient descent algorithm, for its simplicity to implement and compute [11].

$${}^{S}_{E}\mathbf{q}_{k+1} \;=\; {}^{S}_{E}\hat{\mathbf{q}}_{k} \;-\; \mu\frac{\nabla f({}^{S}_{E}\hat{\mathbf{q}}_{k}, {}^{E}\hat{\mathbf{d}}, {}^{S}\hat{\mathbf{s}})}{\|f({}^{S}_{E}\hat{\mathbf{q}}_{k}, {}^{E}\hat{\mathbf{d}}, {}^{S}\hat{\mathbf{s}})\|}, \quad k = 0, 1, 2\ldots n \tag{5.5}$$

$$\nabla f({}^{S}_{E}\hat{\mathbf{q}}_{k}, {}^{E}\hat{\mathbf{d}}, {}^{S}\hat{\mathbf{s}}) \;=\; \mathbf{J}^{T}({}^{S}_{E}\hat{\mathbf{q}}_{k}, {}^{E}\hat{\mathbf{d}}) f({}^{S}_{E}\hat{\mathbf{q}}_{k}, {}^{E}\hat{\mathbf{d}}, {}^{S}\hat{\mathbf{s}}) \tag{5.6}$$

where the symbol $\mathbf{J}^{T}$ represents the transpose of Jacobian. In this scenario, a single iteration is sufficient to give an accurate estimate [11]. The calculations are further simplified by considering the magnetic field of the earth as having a vertical and horizontal component only, and the gravity as only the vertical.

(ii) The other method is to estimate using the gyroscopic data, given that the initial conditions are known. The estimation is done using the angular rate, which is obtained by quarternion product defined by:

$${}^{S}_{E}\dot{\mathbf{q}}_{\omega,t} \;=\; \frac{1}{2}{}^{S}_{E}\hat{\mathbf{q}}_{est,t-1} \otimes {}^{S}\omega_{t} \tag{5.7}$$

$${}^{S}_{E}\mathbf{q}_{\omega,t} \;=\; {}^{S}_{E}\hat{\mathbf{q}}_{est,t-1} \;+\; {}^{S}_{E}\dot{\mathbf{q}}_{\omega,t}\Delta t \tag{5.8}$$

where ${}^{S}\omega_t$ is the angular rate and ${}^{S}_{E}\hat{\mathbf{q}}_{est,t-1}$ is the previous estimation

Now, using these two estimations, they are fused into one output, using the fusion filter:

$$ {}^{S}_{E}\hat{\mathbf{q}}_{est,t-1} \;=\; \gamma_t {}^{S}_{E}\hat{\mathbf{q}}_{\nabla,t} \;+\; (1-\gamma_t){}^{S}_{E}\mathbf{q}_{\omega,t}, \quad 0 \leq \gamma_t \leq 1 \tag{5.9} $$

γ_t is a pre-synthesised weight, based on the sampling interval and divergence rates [11].

We convert these findings into the Euler angle representation:

$$ \psi \;=\; Atan2(2q_2q_3 \;-2q_1q_4,\; 2{q_1}^2 \;+\; 2{q_2}^2-1) \tag{5.10} $$

$$ \theta \;=\; -sin^{-1}(2q_2q_4 \;+\; 2q_1q_3) \tag{5.11} $$

$$ \phi \;=\; Atan2(2q_3q_4 \;-2q_1q_2,\; 2{q_1}^2 \;+\; 2{q_4}^2-1) \tag{5.12} $$

where $q_1 \ldots q_4$ are quaternion scalars, as depicted in Equation 5.1. The Euler angles are corrected for their roll overs [13] using appropriate calculations. Although we have two representations, neither of them is the best [14]. They have their own advantages and drawbacks, and their usage is application dependent.

If Quarternions give us the unit vector axis, about which we must rotate our sensor axis to get the reference axis, the Euler angles are the angles by which we must rotate each of the axes of the sensor to get the axis of the reference frame (earth).

Next we change this to exponential mapping. The exponential mapping for quaternions is given by [15]. This is done to further amplify the differences between the gestures, so that the recognition becomes easier.

The employed algorithm was successful in characterising the movement of the drumstick. The data from the orientation filter was used to make a visual representation, which dynamically showed the movement of the stick. Interestingly there was no latency between the video and the actual movement, owing to the lightness of the algorithm and the super fast computability of max. The video was done using the jitter tool, that comes as a bundle with max. Thus the new interface became as shown in Figure 5.3

Given that we have an efficient representation for the orientation of the stick, we are yet to actually algorithmically recognise the gestures. The Gesture Follower, which is a maxpatch developed by Institut de Recherche et Coordination Acoustique/Musique (widely known as

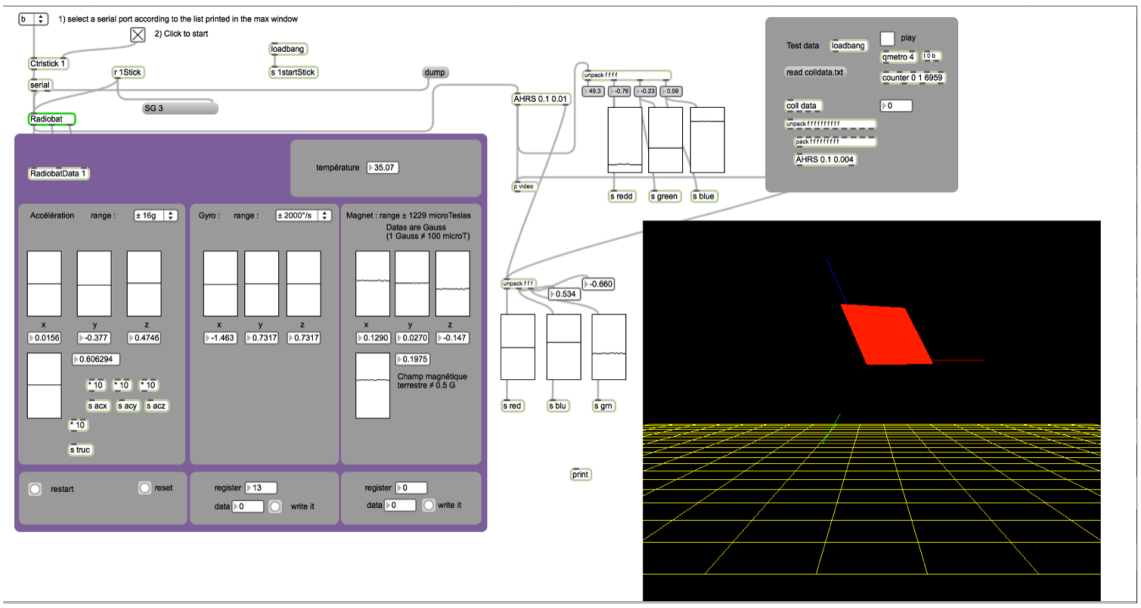

Figure 5.3: The new interface, showing a visual of the Stick's Orientation

IRCAM), Paris. The system is based on Hidden Markov Models. Its present implementation can recognise up to three gestures and is based on two dimensional inputs. The main advantage of this tool is that, it also recognises the time signature of the gesture. So, there is a descriptor that indicates the time index as the learnt gesture is being remade.

The catch with Gesture Follower, however, is that it best works with the data in a normalized range of [0,1]. Even though we have a module in MAX called scale to easily change a given range into another, it was soon realized that mapping was not efficient. So, instead of actually scaling down, it would be better to actually have the data manipulated into that range. This is also the reason that we include exponential mapping of quaternion in our algorithm.

The next challenge faced is the variability of the data. Since we have a fine representation of orientation and due to its relative nature, we are bound to have different data values. This was overcome by feeding a differentiated input. A differentiated input only feeds the rate of change of values, which is going to be the same for a gesture, disregard of the initial values. At the time of submitting this report, we are able to distinguish three simple and totally irrelevant gestures: 1) Circle; 2) Roll; 3) Pull-push along the stick axis;

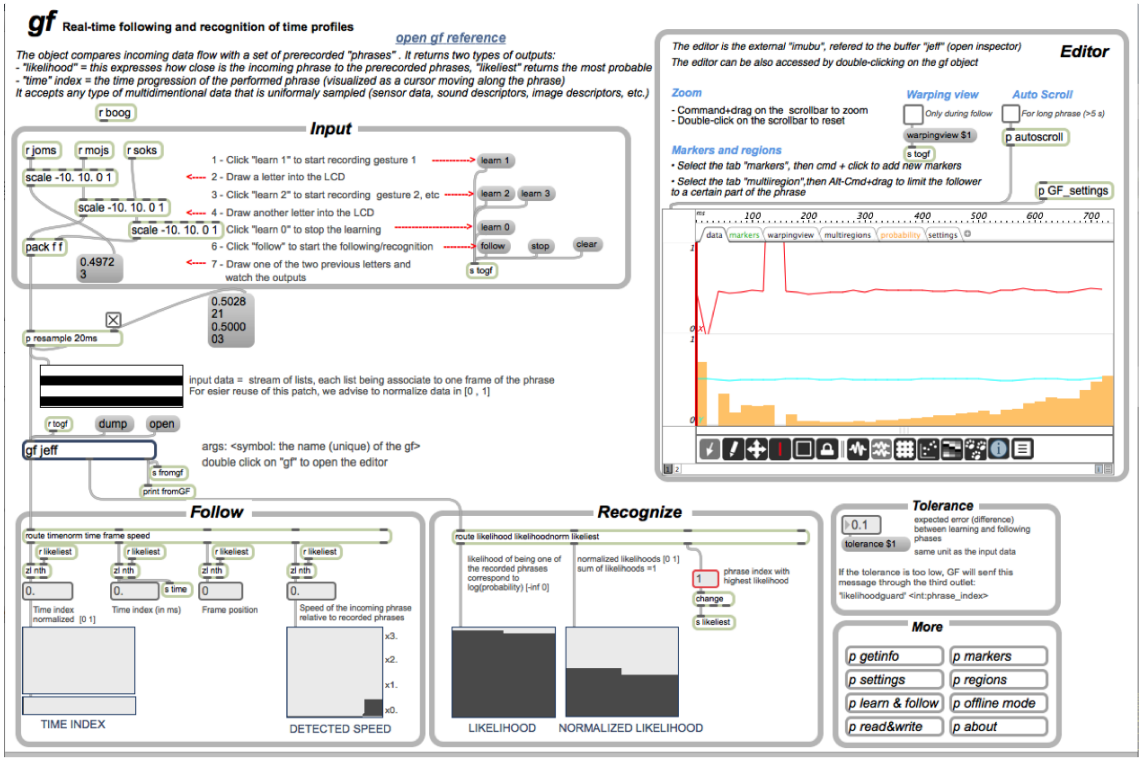

Figure 5.4: The Gesture Follower by IRCAM (has been slightly modified by us for the work)

Conclusion

Chapter 6

Conclusion

The project work comprised of a thorough study to create a novel computer instrument. Although the initial goals were not achieved completely, the research has given an in depth analysis and a solid base to further progress with the work. We can now characterise an impact made with the stick and also recognise simple gestures. The future work is to include more gestures and to control all the top level(refer section on tapong) parameters of the synthesiser. We also aim at making the instrument commercially available in the near future.

This internship was a great learning experience to me, and has given a solid foundation for research. I got deeper insight into how research is actually done and on how to manage the workload. The programming language Max/Msp was learnt and subsequently Latex was also learnt in this report writing process. I have provided sufficient analysis to carry the project to the next level of detecting complex gestures. A link was established between the synthesiser and the sensor system and an appreciable controllability was achieved. The double impact recognition was an original idea. Quaternion is a relatively new concept and its knowledge has given us the ability to use it in other future processes. I also learnt how to find the right algorithm or methodology that is coherent to our work and to frame the solution for the given problem statement. The extensive work has not only given us the directions to work on, but also the directions in which not to proceed. The project work did not achieve its initial aim. It was a partial success. However, the internship was a success, as the aim was to understand research and to learn and implement.

Bibliography

[1] E. Edmonds, D. Everitt, M. Macaulay, and G. Turner, “On physiological computing with an application in interactive art,” *Interacting with Computers*, vol. 16, pp. 897–915, Oct 2004.

[2] P. R. Cook, “Remutualizing the instrument: Co-design of synthesis algorithms and controllers,” ser. Stockholm Music Acoustics Conference, August 2003.

[3] P. Cook, “Principles for designing computer music controllers,” 2000.

[4] E. R. Miranda and M. M. Wanderley, *New Digital Musical Instruments: Control and Interaction beyond the keyboard*. Middleton, Wisconsin: A-R Editions, Inc., 2006.

[5] C. Dobrian, “Aesthetic considerations in the use of “virtual” music instruments,” 2001.

[6] S. Conan, E. Thoret, M. Aramaki, O. Derrien, C. Gondre, R. Kronland-Martinet, and S. Ystad, “An intuitive synthesizer of continuous interaction sounds: Rubbing, scratching and rolling,” *Computer Music Journal*, 2014.

[7] M. Aramaki, C. Gondre, R. Kronland-Martinet, T. Voinier, and S. Ystad, “Imagine the sounds: An intuitive control of an impact sound synthesizer,” in *Auditory Display, Lecture Notes in Computer Science*, ser. CMMR/ICAD 2009. Springer-Verlag, May 2009, pp. 408–421.

[8] M. Mathews and J. O. Smith III, “Methods for synthesizing very high Q parametrically well behaved two pole filters,” 2003.

[9] InvenSense Inc, “Mpu-9150 product specification,revision 4.3,” 2013.

[10] Harvey Weinberg, “Calibrating imems gyroscopes,” 2009.

[11] S. O. Madgwick, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays,” April 2010.

[12] J. M. Cooke, M. J. Zyda, D. R. Pratt, and R. B. Mcghee, “Npsnet: Flight simulation dynamic modeling using quaternions,” *Presence*, vol. 1, pp. 404–420, 1992.

[13] P. B. Jean-Christophe Lementec, “Recognition of arm gestures using multiple orientation sensors: Gesture classification,” in *Auditory Display, Lecture Notes in Computer Science*, ser. IEEE conference on Intelligent Transportation Systems. IEEE, Oct 2004, pp. 965–970.

[14] P. B. Ronan and R. Boulic, “Parametrization and range of motion of the ball-and-socket joint,” *Deformable Avatars, The International Federation for Information Processing*, vol. 68, pp. 180–190, 2001.

[15] F. S. Grassia, “Practical parameterization of rotations using the exponential map,” *Journal of Graphics Tools*, vol. 3, pp. 29–48, 1998.