Guidance, control, and navigation of a quadrotor choreography for a real-time musician-in-the-loop performance

Michael El-Jiz

A Thesis

in the department of

Electrical and Computer Engineering

Presented in partial fulfillment of the requirements

for the degree of

Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

June, 2015

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Michael El-Jiz**

Entitled: **Guidance, control, and navigation of a quadrotor choreography for a real-time musician-in-the-loop performance**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. R. Raut Chair

Dr. B. Vorn Examiner

Dr. K. Skonieczny Examiner

Dr. L. Rodrigues Supervisor

Approved by _____
Chair of Department or Graduate Program Director

_____
Dean of Faculty

Date **June 30, 2015**

Abstract

Guidance, control, and navigation of a quadrotor choreography for a real-time musician-in-the-loop performance

Michael El-Jiz

This thesis proposes a systematic methodology for the guidance, control, and navigation, of a quadrotor to perform a choreographed dance in real-time as a function of the music performed by a musician.

The four main components of a human choreography (namely the notions of space, shape, time and structure) are analyzed and mathematically formulated for a robotic performance. This allows for a real-time interaction with a musician without prior knowledge of the music, and based on the pitch of the acoustic signal. A novel approach for mapping music features to trajectory parameters is proposed, as well as the design of a trajectory shaping filter based on two coefficients that are set in real-time by an artist through a MIDI foot-pedal board. The two coefficients are inspired by a mathematical description of acoustic signals. The proposed approach maps motion parameters and the music to trajectory motifs that are then switched in harmony with the music chord structure. The mathematical formulation of a quadrotor choreography is simulated. The simulation relies on the linearized dynamics and the physical properties of a quadrotor, and produces a graphical representation of the quadrotor choreography. To validate the control system, the position of the quadrotor is compared with the desired position. To measure the effectiveness of the link between music and the position of the quadrotor, the trajectory generator system is inverted to generate a sequence of music pitches. The melodic phrase generated by the position of the quadrotor is played back to the musician. A real-time musical interaction occurs between the musician and the quadrotor. Simulation results show that the proposed methodology yields an effective real-time performance for a quadrotor choreography.

Acknowledgements

My thesis would not have gone past draft -1 without the constant and amazing help of my supervisor, Dr. Rodrigues. You were very patient with me. Second on my thanks-list would be Michael Di Perna, my colleague and friend. I also want to thank (and say "hi!" to) all the students I met that worked under the supervision of Dr. Rodrigues in the HYCONS Lab: Jesus Villaroel, Julia Ghorayeb, Alexander Botros, Ronnilson Rocha, Miad Moarref, and all the others. You have all helped me, even if you think you didn't. You too, undergrads in the flight simulator lab!

Thank you Patrick Edwards-Daugherty and Caroline Glass, from Pleiades Robotics Inc., for your selfless help and support, for your artistically relevant input, and for Dallas.

Ian Hattwick's (from CIRMMT - McGill) expertise in Digital Musical Instruments creatively extended this project. Thanks Brad Luckhart (from the fluids lab at Concordia University) - the results might not be relevant to this thesis, but you helped me concretize some fluids concepts.

I guess my family and friends are also to thank - without them, I probably would have finished earlier. Just kidding, love you.

"The fundamental forms which occur in the decorative arts of all ages and races – for instance the circle, the triangle, the spiral, the parallel — are known as motifs of design. They are not art 'works', not even ornaments, themselves, but they lend themselves to artistic creation. The word motifs bespeaks this function: *motifs are organizing devices that give the artists imagination a start, and so 'motivate' the work. They drive it forward and guide its progress.*"

- Susanne Langer

Contents

List of Figures

1. e4

Chapter 1

Introduction

An Unmanned Aerial Vehicle (UAV) is an aircraft without a human pilot aboard. The concept of UAVs is not a new one: they first appeared as military weapons in the mid 1800s when Austrians used unmanned, bomb-filled balloons to attack Venice [1], where a fuse on-board the balloons dropped the bombs at an *a priori* calculated time. In a non-military context, the use of UAVs can save time, money, and most importantly human lives, by performing remotely-monitored or autonomous tasks. Their tasks include but are not limited to traffic surveillance, road conditions and emergency response [2], forest fire monitoring [3], building inspections [4, 5], package delivery with Amazon's Prime Air [6], and many more.

This thesis will focus on fixed-rotor quadrotors, which are a sub-category of UAVs. As the name suggests, they are characterized by having four fixed rotors. Quadrotors are mechanically simple; they consist of a frame, batteries, four rotors, four motors, four motor controllers, four propellers, an Inertial Measurement Unit (IMU), a radio module and a processor. The quadrotor is controlled by varying the thrust generated by the four propellers. The thesis will specifically focus on the mathematical formulation of a choreography for the guidance and the navigation of a quadrotor, as a function of the pitch of acoustic signals, in real-time.

This chapter is an introduction to the thesis. Section 1 will present the motivation behind the research. The system overview is presented in Section 2. Section 3 will cover the literature survey. The structure of the thesis is presented at the end of this chapter, in Section 4.

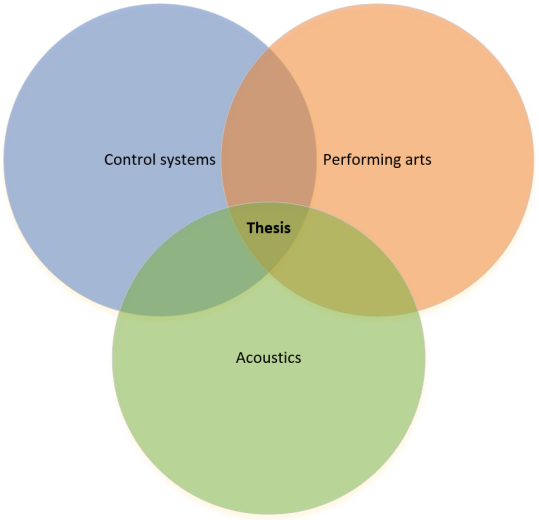


FIGURE 1.1: Venn Diagram representing the work of this thesis, at the intersection of art, acoustics, and control systems

1.1 Motivation

The presence of quadrotors in consumer electronics is increasing. More than a dozen of quadrotors (sometimes called drones) were presented at the Consumer Electronic Show 2015 in Las Vegas [7]. Their piloting is limited to predefined maneuvers, predefined tasks, or to the direct control of the navigation parameters from remote control of a pilot. This allows for precise maneuvering using knobs, buttons and control sticks. However, UAV systems have never been designed for an artist-in-the-loop performance. Traditionally, the navigation parameters of a UAV are the heading angle, the altitude (or the height), and the speed.

An artistic performance is a way for artists to convey their artistic expression through their body or through objects. A robotic artistic performance, by extension, would convey artistic expression through or with a quadrotor.

It is the opinion of the author that humans will interact with quadrotors through human-understandable ways, such as music. The extraction of the pitch of a sound has been an active area of acoustic research for more than 30 years. However, to the best of the author's knowledge, this expertise has not been directly translated to quadrotors for real-time artistic applications, as discussed in the literature survey.

The motivation behind this research is the combination of control systems, performing arts, and acoustics, in a multidisciplinary research (fig 1.1). The system inherits stability from control systems, applies acoustic models in the trajectory design, and is supervised by an artist-in-the-loop.

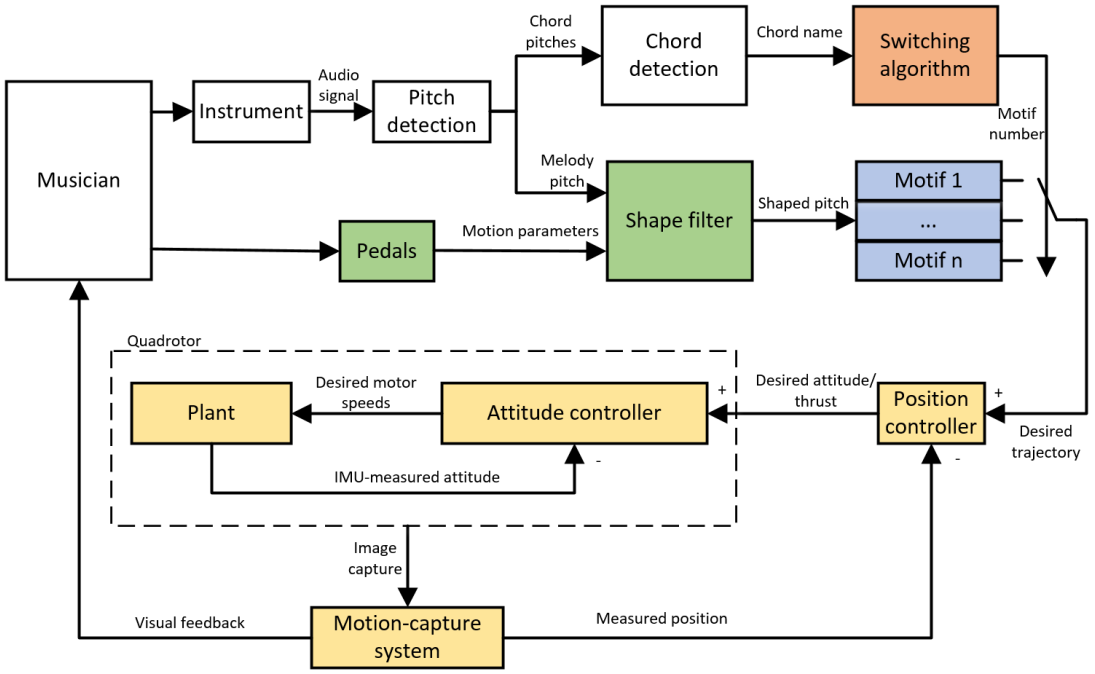


FIGURE 1.2: System overview. The musician block is the supervising artist-in-the-loop.

What are the specifications of a quadrotor system that allow for real-time control with an artist-in-the-loop? How can the basic components of a human choreography be mathematically translated to quadrotor commands for a musician-quadrotor choreography? These are the two main questions to which this research will find answers.

1.2 System Overview

Figure 1.2 presents an overview of the proposed methodology in a hardware-based implementation. This thesis proposes a mathematical formulation of choreography components for performing arts applications with quadrotors, reacting in real-time to the pitch of acoustic musical signals. The system consists of a musician interacting with a quadrotor. During an artistic performance, the musician plays his instrument to generate acoustic signals. The required acoustic features are extracted. A trajectory generator and a trajectory filter transform the parameters into a trajectory, which is then forwarded to the quadrotor control system. Each choreography component is represented by a different color in figure 1.2. For the purposes of this thesis, the system is simulated on a computer.

The proposed combination of art and engineering allows for a stable, dynamic, and eloquent performance, while providing artists the tools to control quadrotors, design a choreography for quadrotors, and to integrate them in a real-time artistic performance.

1.3 Literature Survey

This thesis focuses on the choreography and synchronicity between real-time inputs and quadrotor control. Therefore, a background in robotic choreography and quadrotor control is presented. The literature survey focuses on both parts of the methodology: the art and the engineering. A brief overview of available feature extraction algorithms for the recognition and parameterization of music is also presented. The main components and ideas of designing a choreography are detailed in Chapter 2.

1.3.1 Music feature extraction

This section presents a literature survey of the extraction of the features required for the navigation of the quadrotor. The pitch of a sound is an important notion for both speech and music recognition. Some beat extraction algorithms are presented.

The pitch of a sound is associated with the fundamental frequency of the sound. The pitch of a sound is not a physical property, but a subjective psycho-acoustical attribute of sound. Contrary to popular misconception, the pitch of a sound is the fundamental frequency, but it is not always the frequency component of the sound with the highest amplitude; all the different frequency components of a sound are integer multiples of the fundamental frequency [8].

The pitch of a sound can be extracted through frequency-based or time-based approaches. The standard approach to frequency analysis of a signal is the Fourier Transform. The Fast Fourier Transform (FFT) can be used in real-time digital applications to extract the frequency components of a sound [41]. The principles of harmonic product spectrum, cepstrum or maximum likelihood [36–39], can be applied to find the fundamental frequency of a sound.

The problem of pitch detection has also been solved from a time-based approach. Reference [32] uses a sharp band-pass filter bank at several frequencies to corner the fundamental frequency of the sound. A resonnant-filter approach is used in [40] to extract the frequency components of sounds, and can be used to extract the pitch of sounds. This method, based on a bank of resonnant filters, is computationally expensive, and requires fine parameter tuning. The use of auto-correlation also allows for fundamental period extraction [22]. Dynamic Programming (DP) and Artificial Neural Networks (ANN) are used in [33] to extract the pitch. The positive zero crossings for each voiced segment are determined, then the search space of pitch period hypotheses is generated, and finally the best path is computed. The period hypotheses along this path represent the sequence of pitch periods within the voiced segment. Another probabilistic

approach is proposed in [35] through the use of a temporally-constrained convolutive models. Reference [34] detects the pitch through an envelope estimator. The topic was also addressed in references [23, 31] where different algorithms to detect the pitch of a sound were compared.

According to [20], rhythm is "the aspect of music concerned with the organization [of sounds and silences...] in time". Therefore, besides responding to the pitch of a musical piece, rhythm recognition is a very important task in the control of robots in response to music. The beat-tracking algorithm used in [18, 19] is proposed in reference [27], which describes a real-time audio-based beat-tracking system for music with or without drums. This method is based on multi-agent beat predictors with a score assignment approach. Reference [21] proposes a tempo and beat analysis of music through a network of resonnators to phase-lock with the beat of the signal. As an update, reference [24] uses simpler filters and a more heavily processed signal. The beat-detector works by passing the signal through band-pass filters, extracting and processing the envelope of each band. The envelopes are then passed through a bank of comb filters such that a filter with delay matching the period of a pulse train will have larger output than a filter with mismatched delay. The outputs are summed across frequency subbands and the energy output of these filters will reveal the strongest periodic component of the signal. This approach, similar to [40] for pitch extraction, requires fine-tuning of the parameters.

1.3.2 Robotic choreography and music, and quadrotor control

A choreography is the sequence of steps and movements in a dance, usually performed by human dancers. By extension, choreography has been defined in a robotic sense as the artistic motion of a robot [75]. An artistic robotic performance is an approach for an artist to express his or her idea through the use of a robot. The artist controlling the robot is "twice removed" from the performance (artist to computer, computer to robot), and therefore the conveyability of emotions is reduced in a traditional sense, while new and different freedoms of expression are granted to the artist [75].

References [18, 19] present a humanoid robot that tracks real-time musical cues through an on-board microphone and dances (sways side-to-side) and sings with the music. However, the robot was simply comparing the detected tempo with that of songs in a limited database to find the matching song it was listening to. Robots have also been taught to play music. Reference [65] presents an interactive improvisational robotic marimba player that responds to a pianist's cues, with preset and improvisational musical phrases.

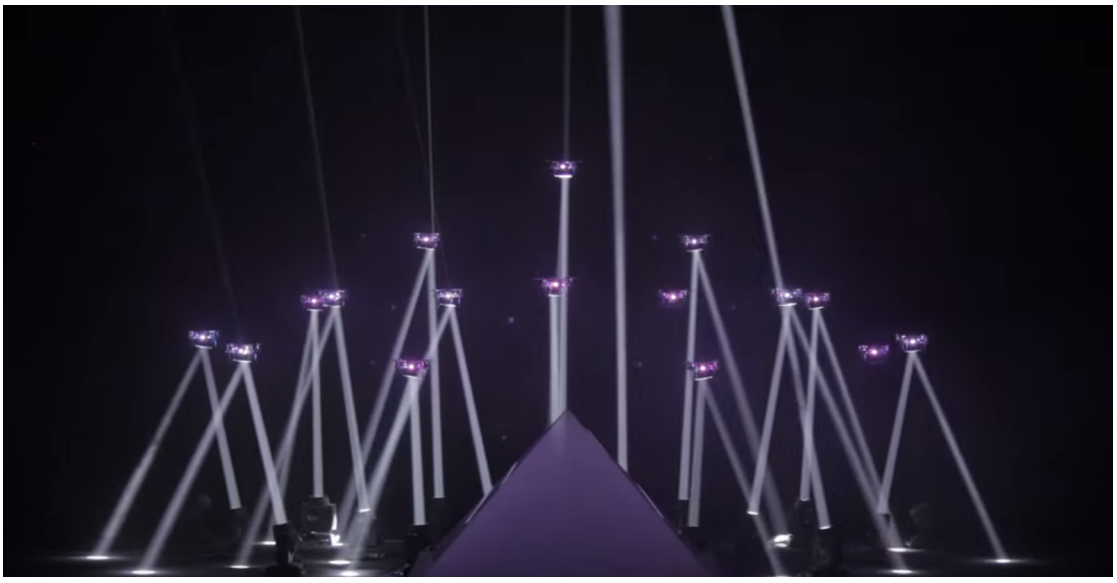


FIGURE 1.3: Quadrotor Artistic Performance [69]

Other instruments have been played by robots, such as bagpipes [64], theremin [66], violin [67], piano [68], to name a few.

The integration of UAVs in artistic performances has only been studied during the last five years. The pioneering work in reference [9] appears to be the first paper on integration of quadrotors with music choreography. The paper addresses the control of a quadrotor that sways to the beat of a music piece with offline beat detection. The problem is not one of music extraction and real-time mapping since the music feature extraction happens offline, but rather a control problem of a quadrotor following a time-sensitive periodic trajectory. This limits the musical interaction and does not allow for musical improvisation by a human player in real-time, or new maneuvers from the robot. This work has later been extended in reference [10] where a framework that identifies feed-forward parameters for precise periodic quadrotor motions was introduced. An extension to the work [97] describes a quadrotor choreography mathematically, based solely on the beat of the music, following periodic trajectories. Naturally, the trajectories are generated through a Fourier series, and the coefficients are designed to create the desired effect. Figure 1.3 taken from reference [69] displays a multi-quadrotor performance using lights and mirrors. Reference [29] is an artistic project combining lamp shades and quadrotors. These performances are pre-programmed and do not allow for human input. The approach in [12] differs by having the quadrotor create music by flying and interacting with instruments. However, this is also prepared, pre-programmed, and does not allow for human-robot interaction. In reference [76], three types of paths serve as a trajectory generation basis, depending on pre-identified melodic phrases.

As mentioned, fixed-rotor quadrotors are mechanically simple: their motion is controlled

by adjustments to the individual rotational speeds of four propellers. The direction of the total thrust with respect to the world frame is managed by controlling the orientation of the quadrotor.

Throughout the literature, modeling the quadrotor reveals similar system dynamics with different properties (such as the mass, the size, and the moments of inertia of the UAV). However, different control algorithms can be applied to stabilize the quadrotor. Reference [77] models the quadrotor along with aerodynamical effects and utilizes a Proportional-Derivative (PD) controller. However, the gains of the controller are acquired through trial and error. Reference [78] uses integral backstepping to stabilize the UAV. The same author in reference [79] compares the performance of a Proportional-Integral-Derivative (PID) controller with that of a Linear Quadratic (LQ) regulator - the latter performs poorly due to model imperfections.

1.4 Contributions

The main contribution of this thesis is the proposition of a systematic methodology for the control of a quadrotor robot based on acoustic musical signals in real-time. The goal of the proposed approach is to give an artist the tools to control and choreograph a quadrotor performance in real-time. To streamline the research effort, the thesis will focus on controlling a quadrotor based on the pitch of the acoustic music signal. Possible extensions to other features will be briefly covered in Chapter 5. As a proof of concept, the system has been implemented in a simulation. The following are the detailed contributions of this thesis:

1. The four main components of a human choreography (namely the notions of space, shape, time and structure) are analyzed and mathematically formulated for a robotic performance. To the best of the author's knowledge, the robotic choreography problem has never been approached for a real-time interaction with a musician without prior knowledge of the music, and based on the pitch of the acoustic signal.

2. A novel approach for mapping music features to trajectory parameters is proposed, as well as the design of a trajectory shaping filter based on two coefficients that are set in real-time by an artist. The two coefficients are inspired by a mathematical description of acoustic signals.

3. The mathematical formulation of a quadrotor choreography performance is simulated in Python. The simulation relies on the linear dynamics and the physical

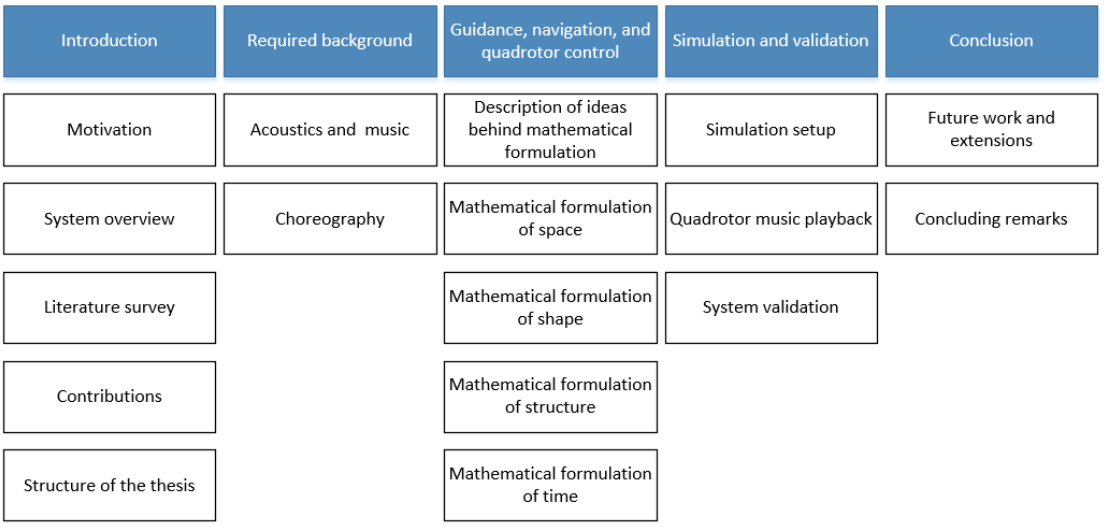


FIGURE 1.4: Map of the thesis

properties of a quadrotor, and produces a graphical representation of the quadrotor choreography. As a validation, musicians can control the simulation with their musical instrument to design a quadrotor choreography.

4. To validate the control system, the position of the quadrotor is compared with the desired position. To measure the effectiveness of the link between music and the position of the quadrotor, the trajectory generator system is inverted to generate a sequence of music pitches. The melodic phrase generated by the position of the quadrotor is played back to the musician. Therefore, a real-time musical interaction between the musician and the quadrotor occurs.

1.5 Structure of the Thesis

Chapter 2 presents the required background in music, choreography composition, and dance. Chapter 3 presents all the mathematics leading to the trajectory generation, the dynamics of the quadrotor, and the controllers based on the linearized dynamics. This is the theory behind the quadrotor control subsystem of 1.2 (contributions 1 and 2). Chapter 4 discusses the simulation setup and validates the results (contributions 3 and 4). Chapter 5 concludes the thesis and presents possible future work. Figure 1.4 is a graphical representation of the structure of the thesis.

Chapter 2

Background in music and choreography

This chapter presents notions in music and choreography required to understand the work developed in this thesis. It explains the notions of pitch and chords. This chapter also discusses the main components of a human dance composition.

2.1 Music

Music is an art form based on sound. It has been theorized that music has been around since the Palaeolithic era (around 35,000 years ago); archaeologists found flutes carved from bones in which lateral holes had been pierced [90]. According to [92], music is an inherent and universal human quality. This quality can tentatively be translated to robots by extracting the necessary musical features and turning them into parameters the robot could interpret. In order to parametrize music, one must understand the basic components that define a melodic phrase.

2.1.1 Music theory

The pitch of a sound generated by a musical instrument is the fundamental frequency of a sound. The pitch is usually associated to a note. Notes go from A to G in what are called tone intervals, except for the intervals $B-C$ and $E-F$ which are semitones. An octave contains 12 notes in total. Geometrically, the pitch space of an octave can be represented by a circle, which is a one dimensional manifold (see Fig. 2.1 and reference [80]). The same notes repeat themselves every octave, but at fundamental frequencies

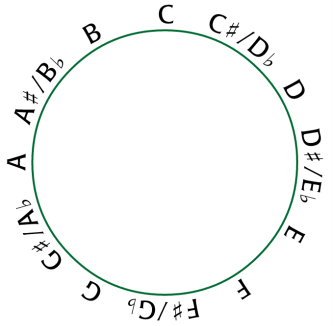


FIGURE 2.1: Circular pitch class space

that are doubled (if the octave is higher) or halved (if the octave is lower). For example, the note A_4 (the note A on the 4^{th} octave) has a fundamental frequency of 440Hz. The fundamental frequency of A_5, the A an octave higher, is 880Hz. A melodic phrase is a combination of notes or pitches held for different times in a specific order.

2.1.2 Pitch Detection

The pitch is a psycho-acoustic property. The human ear can reconstruct that missing frequency from the harmonic components of a sound. To extract the pitch information from the physical sound created by a musical instrument, signal processing is performed on the captured sound. The fundamental frequency - or the pitch - of a sound is not necessarily the frequency with the highest amplitude. For most acoustic instruments, finding the fundamental frequency equates to finding the highest frequency peak in the spectral representation of a sound.

Any sound can be represented by the sum of its frequency components [42]. The Fourier transform is a mathematical procedure that decomposes a signal into a function of frequency. The Fourier transform $F(\omega)$ of a function $f(t)$ can be found through equation (2.1), where f is a real number that represents frequency.

$$F(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi j t f} dt \tag{2.1}$$

The discrete equivalent of the Fourier transform, the discrete Fourier transform, converts a finite sequence of equally spaced samples of a function into the sequence of coefficients of a finite combination of complex sinusoids, ordered by their frequencies. The discrete Fourier transform of a signal $x[n]$ with N samples can be obtained as:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-2\pi j k n / N} \tag{2.2}$$

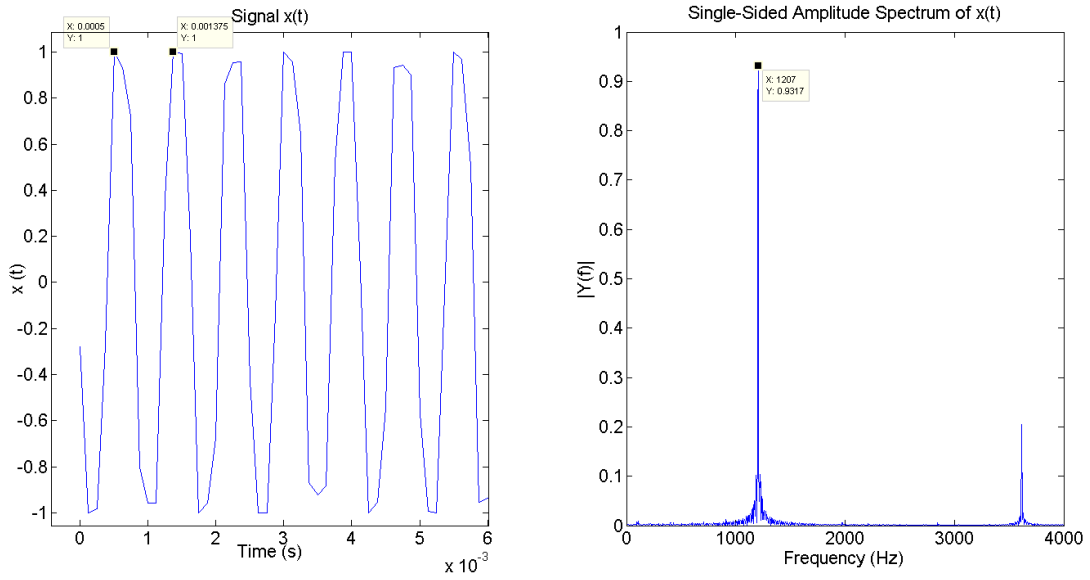


FIGURE 2.2: Fast-Fourier transform of a 1205Hz tone.

In equation (2.2), the frequency ω_k can be retrieved through equation (2.3), where F_s is the sampling frequency.

$$\omega_k = 2\pi k Fs/N; \tag{2.3}$$

The fast Fourier transform (FFT) is a method that efficiently computes the discrete Fourier transform. The FFT requires the number of samples to be a multiple of 2 [60]. It can be utilized to recognize the frequency components of an acoustic signal. Figure 2.2 presents the FFT of a sound. The sound is a 1205 Hz audio signal generated from the *Tone Generator* app on Windows Phone, recorded through the internal microphone of a Samsung Series 9 laptop and processed on Matlab. The sound is sampled at 8KHz and the FFT is run on 1024 samples. The 1205 Hz sound is accompanied by a harmonic component at 3650 Hz.

The resonant filters approach [87] is another frequency decomposition technique. The idea stems from the cochlea: the auditory portion of the inner ear in mammals. The shape of the cochlea allows different frequency components of a sound to stimulate different mechanosensory cells [61]. Figure 2.3 is a drawing of the cochlea; the location of audible frequency resonances are marked along the cochlea. Similar to a filter-bank model of the mammalian cochlea, the resonant filters approach is based on an array of simple physical resonating receivers. Figure 2.4 is a spectrogram comparing two frequency decomposition approaches for the same signal. Higher amplitude components are represented in red, and lower amplitudes are in blue. The resonant filters approach is compared to a simple Fast Fourier Transform. The precision of the resonant filters is

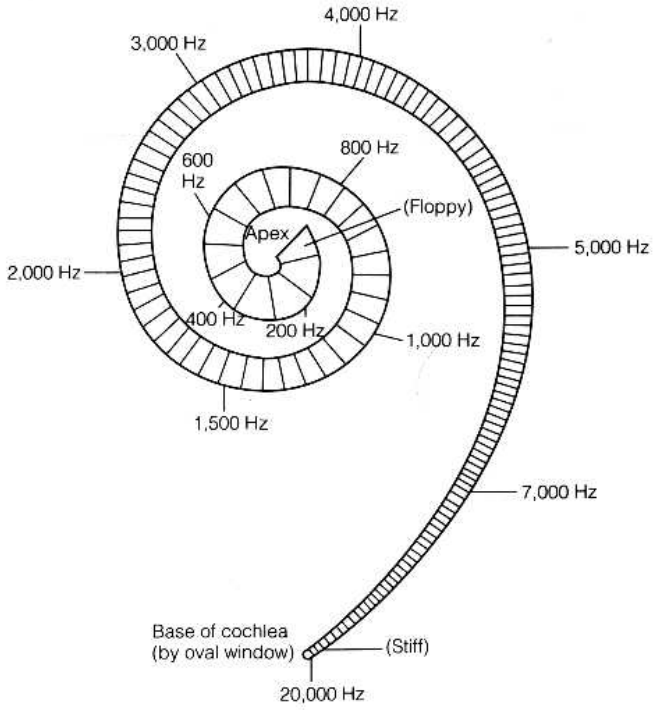


FIGURE 2.3: Drawing of the cochlea and the frequency-location mapping of mechanosensory cells [62].

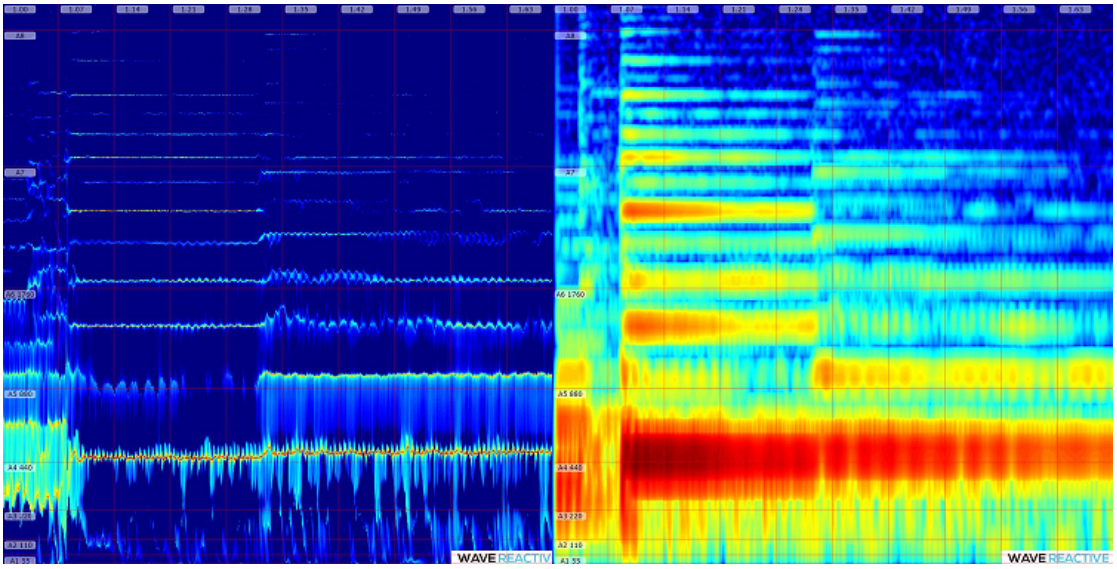


FIGURE 2.4: Comparison of resonant filters (left) and FFT (right) for the frequency decomposition of a sound. Picture from [87]

remarkable. However, the resonant filters are more computationally expensive and rely on the fine tuning of the parameters of the bank of filters.

The following code is a pitch extractor, written in MATLAB, based on the fast Fourier transform:

```
function freq=frequency(signal,Ts)
 \%function freq=frequency(signal,Ts);
 \% This function determines the fundamental frequency of a signal sampled
 \% every Ts seconds

 z=abs(fft(signal));
 noise_level=(max(z)-mean(z))/5;
 n=2;
 while abs(z(n)-z(n-1)) < noise_level
     n=n+1;
 end;
 freq=n/length(z)/Ts;

 end
```

There are several other publicly available approaches at pitch extraction. Max/MSP is a graphical programming language for music processing in which algorithmic blocks are called patches. One could use the MIDI Merlin patch [82] to extract the pitch of a sound from the audible physical signal of a monophonic instrument, in real-time. The pitch is described in the MIDI format. For polyphonic melodies, an alternative to MIDI Merlin on MaxMSP is the PredominantMelody function, part of the C++ open-source audio library Essentia [83].

2.1.3 The MIDI Format and musical representation

The Musical Instrument Digital Interface (MIDI) is a popular standard, used to connect electronic musical instruments and computers. It was developed by a company called *Sequential Circuits* in 1970 (now owned by *Yamaha*) as a standard way to control analog synthesizers [81]. In the MIDI format, a positive integer is associated to each note. Therefore, MIDI information can be stored and processed digitally. The lowest note that the MIDI digital format can represent is a C_{-1}, five octaves below middle C. This note has a fundamental frequency of $8.176Hz$. Let P be an integer representing a note, where $P = 0$ represents the lowest note. The highest note reached by MIDI is a G_9, at $12544Hz$, represented by $P = 127$. Equation (2.4) converts the frequency of a sound to the integer P [70].

$$P = 69 + 12 \log \left(\frac{2\pi f}{440} \right) \tag{2.4}$$

Let O be the octave to which the played note belongs and let $\tilde{P}$ be the pitch class of the note. The mod operator returns the remainder of a division by a certain number. The expressions for the pitch class $\tilde{P}$ and octave O are:

$$\begin{aligned} \tilde{P} &= P \mod 12 \\ O &= (P - \tilde{P})/12 \end{aligned} \tag{2.5}$$

The use of the extracted features from the music into trajectory planning algorithms is discussed in Chapter 3. It is noteworthy to mention that the MIDI format is very popular and includes other information not used in this thesis, such as *pitch-bend*, *program change* and *aftertouch* messages. It also has non-musical applications, such as light control in concerts.

2.1.4 Chord Recognition

A chord is a combination of multiple pitches played at the same time. During musical performances, musicians play chords regularly on polyphonic instruments (musical instruments that can play more than one note at a time). The chord being played is extracted and analyzed in this thesis to create new quadrotor mappings. The most frequent type of chords are triads, chords created by three pitches. The most popular triads are Major, Minor, Augmented and Diminished chords. This section will explain the difference between those chords and will also present a chord recognition algorithm.

An A-Major chord will have a difference of 4 semi-tones between the first note (an A) and the second note (a C#), 3 semi-tones between the second note (C#) and third note (E), and 5 semi-tones between the third note (E) and the first note of the chord on the next octave (also an A). Based on this information we can form what are called difference matrices (Δ) of a chord. For example, the A-major chord is represented by a semi-tone difference matrix of $[4 \quad 3 \quad 5]$. A C-Major chord would have the pitch classes $C-E-G$, and would have the same difference matrix than the A-major chord. A minor chord, for example, would have a difference matrix of $[3 \quad 4 \quad 5]$. Notice that the sum of all entries in the matrices is equal to 12, which is the total number of semi-tones in an octave. The template difference matrices for the four triads are given by

$$\begin{aligned} \Delta_{major} &= [4 \quad 3 \quad 5] \\ \Delta_{minor} &= [3 \quad 4 \quad 5] \\ \Delta_{augmented} &= [4 \quad 4 \quad 4] \\ \Delta_{diminished} &= [3 \quad 3 \quad 6] \end{aligned} \tag{2.6}$$

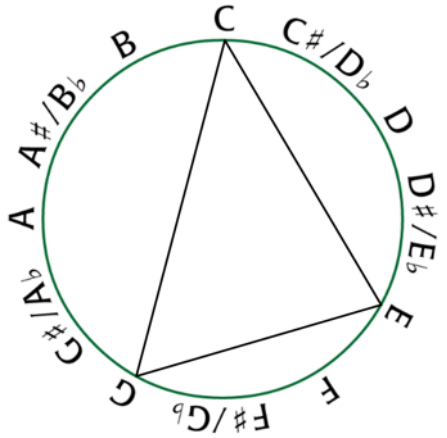


FIGURE 2.5: Circle of notes in green, in which the notes of a C-major chord are connected through black lines.

The use of difference matrices enables triad chord recognition (even for the case where the base note is not the lowest note played, called inverted chords). Appendix A contains C++ code which detects the mentioned triads.

In order to recognize chords, a difference matrix is generated based on the notes played and compared to the template difference matrices. This difference matrix can be represented in the circle of notes. Figure 2.5 is a circle of notes in which the notes that constitue a C-major chord are connected in a triangle. A graphical representation of the difference matrices would be the triangles that form triads. The notes that constitute any major chord are connected by a triangle identical to the C-major chord.

2.2 Choreography

A choreography, derived from the greek words *choria* (circular dance) and *graphia* (writing) is the art of designing a sequence of movements. There are four main notions a choreographer takes into consideration when designing a human dance: space, shape or energy, time, and structure or dynamics [25, 26]. This section will present the main components of a human choreography.

The space of a choreography refers to the space utilized by the dancer(s) during the performance. The space can be studied from a symmetry point of view, by analyzing if the dancers move in a symmetrical or asymmetrical motion. The dancers can move in a straight line, in curved motions, and other geometric forms. These geometric forms are also studied by the choreographer. The scale of the occupied space is also an important aspect of a dance - it refers to the space occupied by the dancer in comparison to the allowable space. Reference [26] describes the space as the "*geometry of space articulated*

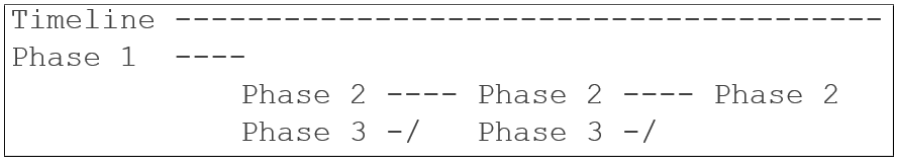


FIGURE 2.6: Progression of a dance as a function of different phases of a dance, as noted by Annable [26]

in terms of 'complex configurations', 'oppositions', 'balance', 'intersecting planes' and 'form', to name but a few concepts".

The shape of a choreography refers to the shape of the dancer and the motion of his/her body. Symmetry and asymmetry of the dancer's body shape can be used to infer stability or instability. The level of a shape refers to the different physical levels that the dancer or dancers reach. As the level changes, the dancers can support themselves on different elements on a stage. The scale of the dance can be explained by the type of motion - such as the contrast between angular, curved, or straight shapes, or the contrast between shrinking and expanding motions, or the difference between contracted and extended body shapes. The shape could also refer to the energy behind specific dance moves. In her notebook, choreographer Annable (reference [26]) describes the shape of 5 dancers as follows:

- Dancer 1 is elongated, smooth and extended
- Dancer 2 is fluid and sustained
- Dancer 3 tends to be fast, sharp and precise in her movements
- Dancer 4 is sharp and cutting
- Dancer 5 has a slow and suspended quality

The timing of the choreography usually accompanies that of the music the dancers are following. The tempo, the metre and the dynamics of the dance are important aspects of the choreography as they usually relate the music to the dance.

The fourth element of dance composition is the structure (or the flow) of a dance. Similar to a song, a dance has different motions, or phases. The different movements of the dancers along the different spaces create a structure through combinations, variations and repetitions. Annable (reference [26]) describes the flow of a dance by the changes in phases of the dance as a function of the main timeline (figure 2.6).

These elements are mathematically formulated and transposed for a quadrotor dance in Chapter 3.

Chapter 3

Guidance, navigation, and quadrotor control

Chapter 2 presented different acoustic feature extraction techniques, and the main components of a human choreography. This chapter provides a mathematical description of the choreography components defined in chapter 2. Each section will cover one of the main choreography components, namely space, shape, structure, and time. Figure 3.1 is a simple block diagram outlining how the musician interacts with the system during the performance.

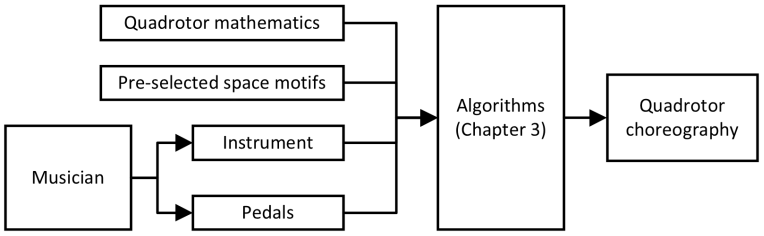


FIGURE 3.1: Choreography simplified block diagram

3.1 Introduction

This section will describe the main components of a quadrotor choreography. As a reminder, there are four main components that constitute a human choreography:

- The notion of space refers to the geographical space occupied by the dancer.
- The shape of a dancer is the physical shape of his or her body and movements.

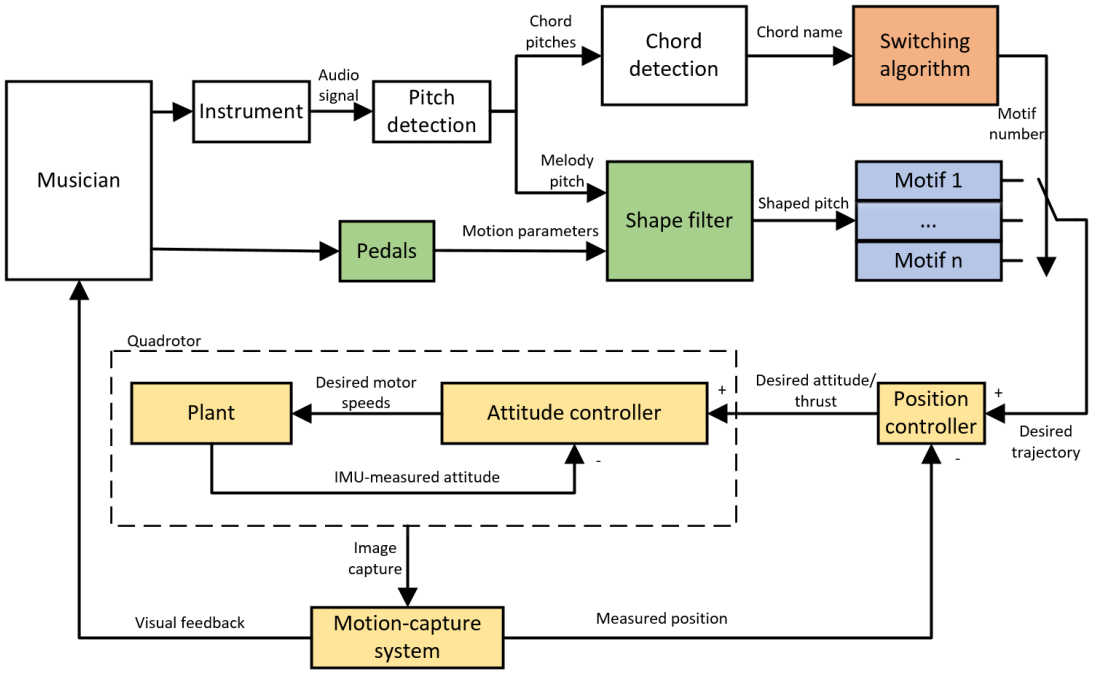


FIGURE 3.2: Block diagram of the system. (Space: blue, shape: green, structure: orange, time: yellow)

- The structure is the succession of the dance moves, the changes in the flow of the dancers, and the general structure of a dance along the main time-line.
- The time component is the timing of the movements and the rhythm of the music and the dance.

In order to make a quadrotor dance, each of these components will be mathematically formulated. Figure 3.2 is a block diagram of the quadrotor choreography system. The blocks that correspond to each choreography component have been highlighted in a specific color: the space is in blue, the shape is in green, the structure is in orange, and the time is in yellow. The overall objective is to use the control inputs (time component) to make the quadrotor follow a trajectory as a function of the music played by a musician. The trajectories are generated along the path motifs (space) and following the motion specifications of the musician (shape). The path motifs can be changed (structure) throughout a performance, in order to add structure to the choreography.

Before describing each of the dance components in more detail, the next section will motivate this discussion by presenting the results of a survey conducted at the Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT) that highlights the features that artists expect to see as a reaction of a robot to a piece of music.

3.1.1 Music moods and survey

There is an undeniable, scientifically-proven relationship between music and emotion [103, 104]. Moods are a manifestation of emotion, and mood classification is also applied to music. There are two recurrent types of representation regarding moods in music. In the first type of representation, the music is categorized according to four or five basic emotions (such as happiness, sadness, fear, anger, and tenderness); emotions are grouped together in clusters (see figure 3.3). In the second type, emotions are spread in a two-dimensional graph as a function of two distinct emotions, for example, valence and arousal. This is called the dimensional model (see figure 3.4) [106]. Music mood classification algorithms have been developed. Reference [106] categorizes the moods and emotions felt by music listeners according to user-generated tags on online music-streaming services such as *last.fm*, while reference [105] categorizes music using audio and lyrics information. Moodswings [102] is a game that harnesses "human computation" for the collection of label data for mood ratings of music. Music Information Retrieval Evaluation eXchange (MIREX) classifies the moods of music in five main music clusters [107]. Each musical mood cluster is described by distinct and precise emotions:

- Cluster 1: passionate, rousing, confident, boisterous, rowdy
- Cluster 2: rollicking, cheerful, fun, sweet, amiable/good natured
- Cluster 3: literate, poignant, wistful, bittersweet, autumnal, brooding
- Cluster 4: humorous, silly, campy, quirky, whimsical, witty, wry
- Cluster 5: aggressive, fiery, tense/anxious, intense, volatile, visceral

A survey has been sent to members of the Center for Interdisciplinary Research in Music Media Technology (CIRMMT) in an attempt to find a correlation between the motion of a robot and the five different MIREX music mood clusters. The complete survey is available in Appendices B and C. Participants were asked to assign an expected quadrotor motion to each music mood cluster. The following options were given as possible quadrotor motions:

- Fast and oscillatory
- Fast and sharp
- Slow and oscillatory
- Slow and sharp

FIGURE 3.3: Categorical model of emotions [94]

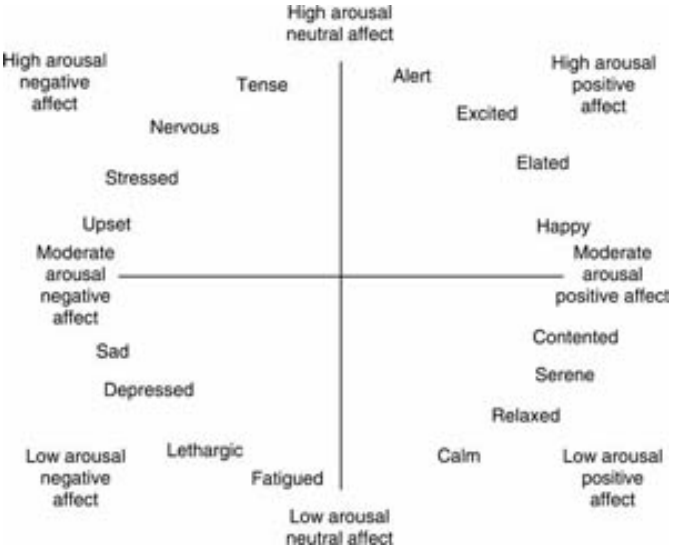


FIGURE 3.4: Dimensional model of emotions [95]

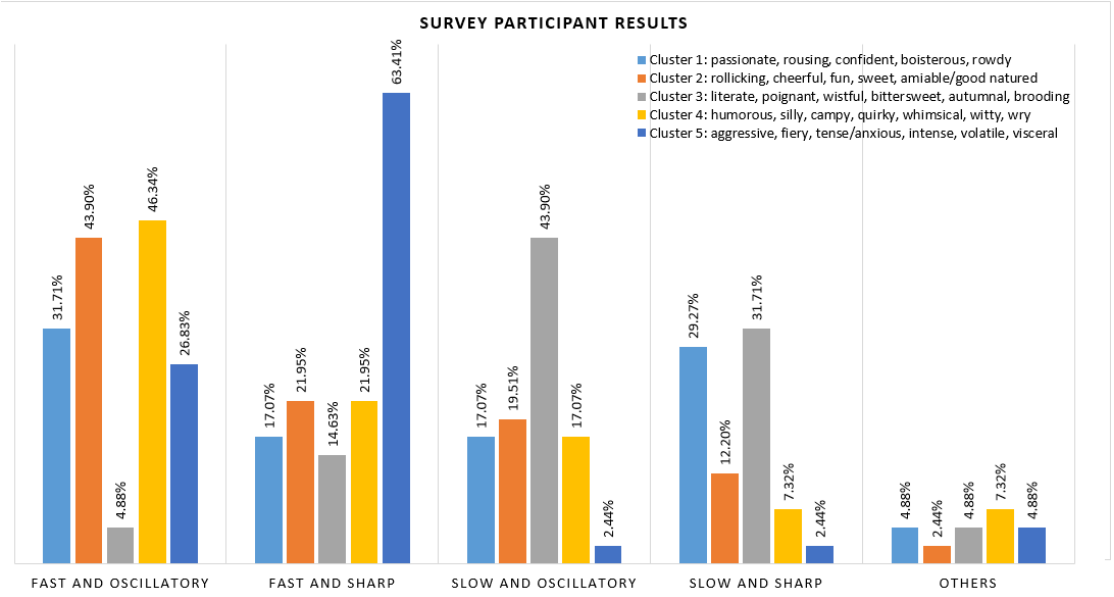


FIGURE 3.5: Survey answers to the MIREX music mood clusters, in percentage

- Other (please describe)

Figure 3.5 presents the answers from the 41 participants who answered “Yes” to the question “Are you a musician, or a performance artist?”. The results are worth discussing. The answers regarding clusters 2 and 4 are very similar to each other. It is important to also notice the similarities in the adjectives used to describe the two clusters. For cluster 3, 75.6% of the 41 participants have chosen a ”slow” motion. For cluster 5, 63.4% of the 41 participants agreed that the expected motion would be ”fast and sharp”. The two contradictory motions “fast and oscillatory” and “slow and sharp” were chosen for cluster 1 at around 30% of the 41 participants. There certainly is some correlation between the music mood clusters and the expected motion of the robot. For music evoking the aggressive mood (cluster 5), the ”fast and sharp” motion is expected. For the happy and humorous emotions (such as cluster 2 and 4), the ”fast and oscillatory” motions are expected. Participants were able to associate robotic motions to certain mood clusters. This shows that the fast-slow and oscillatory-sharp descriptors for robotic motions can be utilized to convey a mood. The final shape is controlled by the artist; it can be chosen to contrast the music or accompany it.

3.1.2 Description of space

For any music being played, the desired position of the quadrotor will be in the space it is assigned. The concept of path motifs is introduced: path motifs are one dimensional manifolds embedded in a three-dimensional space, parameterized by a musical feature.

A path motif serves as a basis for the trajectory generation system. The quadrotor system is designed such that the desired trajectory is along the path motif. The desired trajectory and the path motif are similar to trains and rails: a train can only navigate along its rails - the desired trajectory can only exist along a path motif. The final desired trajectory of the quadrotor is generated from the equations of a specific path motif, and any point of the final desired trajectory will satisfy the equations of the motif it is based on. The idea and the motif designation stems from a quote by philosopher of mind and of art Susanne Langer [26]: "The fundamental forms which occur in the decorative arts of all ages and races – for instance the circle, the triangle, the spiral, the parallel — are known as motifs of design. They are not art 'works', not even ornaments, themselves, but they lend themselves to artistic creation. The word motifs bespeaks this function: *motifs are organizing devices that give the artist's imagination a start, and so 'motivate' the work. They drive it forward and guide its progress.*" The straight line, the circle, and the helix, are presented later as motifs in the mathematical formulation of space.

The quadrotor will be tracking the final desired trajectory through a control system. The desired trajectory is along the path motif. However, it is noteworthy to mention that the quadrotor might not be always moving along the desired path: the quadrotor control system is designed to track the desired trajectory and guarantee convergence, but there are always tracking errors. The results are detailed in Chapter 4.

3.1.3 Description of shape

The quadrotor is a rigid body, and cannot alter its shape. However, the shape of the quadrotor refers to the motion of the quadrotor. The desired trajectory of the quadrotor is within the flying space by generating specific trajectories along a path motif. Any point along the generated desired trajectory of the quadrotor will belong to the path motif, and the equations of the path motifs are used in the trajectory generation. Without a shape filter, the path motifs will generate geographic waypoints as a function of the different integer pitches the musician plays. A shape filter generates trajectories along a path motif by filtering the step pitch input to generate a continuous shaped pitch. Figure 3.6 is a block diagram detailing the function of the shape filter: it generates a continuous shaped pitch from discrete pitches.

The continuous pitch can have different responses, based on the motion parameters (see figure 3.6). Changing the properties of the shape filter will result in changes in the overall trajectory. A fast filter response generates a faster trajectory, and, vice versa, slower responses generate slower trajectories. Different desired quadrotor motions along the path motif can be generated through the shape filter. The musician that is making

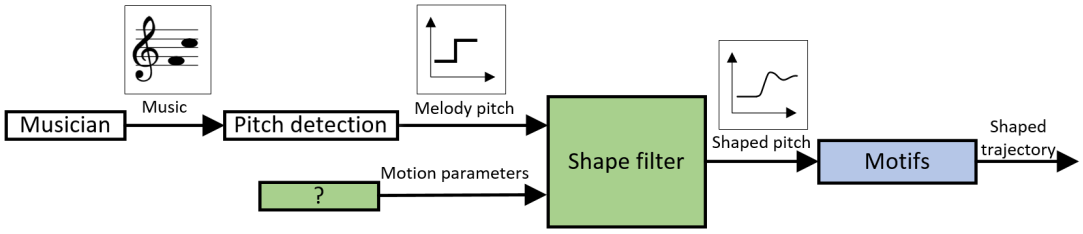


FIGURE 3.6: Shape filter with graphical representations of the signals

FIGURE 3.7: The FCB1010 Behringer MIDI pedal, with 10 button pedals and two expressive pedals [93]

the quadrotor dance can control the speed and the oscillatory nature of the shaped pitch and of the trajectory through two expressive foot-pedals on a MIDI pedal board (see Figure 3.7). The trajectory generation method is based on the physical properties of a sound, which will be described in the next section.

3.1.3.1 Acoustics

A sound is a vibration, and its propagation medium is the air. Biological auditory receptors, namely the ears for humans, capture those vibrations. Each species that is capable of hearing has a different auditory range; the human ear can hear frequencies between 20 Hertz and 18000 Hertz. Naturally, a sound decays over time. There are three important physical properties in a sound [59]:

- The frequency, which is associated to the pitch of a sound, is the most essential component - it is the frequency at which the physical body (a resonator) that generates the sound vibrates.
- The amplitude of a sound is commonly referred to as the volume.
- The decay of a sound describes how fast the resonator stops vibrating.

If k is the starting amplitude of the vibration, ω is the angular frequency of vibration, a is the decay time constant, and ϕ is the phase, a real sound signal A can be modeled by the following equation:

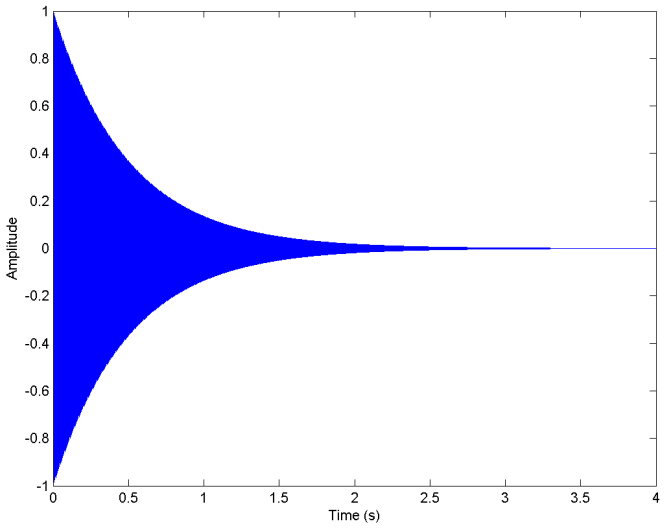


FIGURE 3.8: Amplitude of a MATLAB generated sound as a function of time.

$$A(t) = ke^{at}\cos(\omega t + \phi) \tag{3.1}$$

Figure 3.8 represents the amplitude of a MATLAB [1] generated sound as a function of time, with $\omega \approx 3770\ rad/sec$, $k = 1$, $a = 0.5$, and $\phi = 0$. A trajectory that mimics the physical properties of a sound can be generated. Let us suppose the simplest example: the artist wishes to control the movement of the quadrotor from point A to point B. The points are set, and the path motif is the straight line connecting those two points. By following the physical properties of sound, the amplitude, the frequency, and the decay, are translated into the following trajectory generation parameters:

- The amplitude of the motion is set by the distance between points A and B. The resonator stimulated at $t = 0$ can be associated to the beginning of the movement of the quadrotor.
- The frequency of the resonator is translated to the speed of the motion of the quadrotor as it reaches (or overshoots and oscillates around) the desired waypoint (B) along the path motif.
- The decay rate is related to the damping of the motion of the quadrotor - oscillatory or sharp.

The motion of the quadrotor will therefore be described as oscillatory or sharp to replicate the decay rate of a sound, and as fast or slow to relate to the frequency component in the model of a sound.

[1] MATLAB is a trademark of MathWorks Inc.

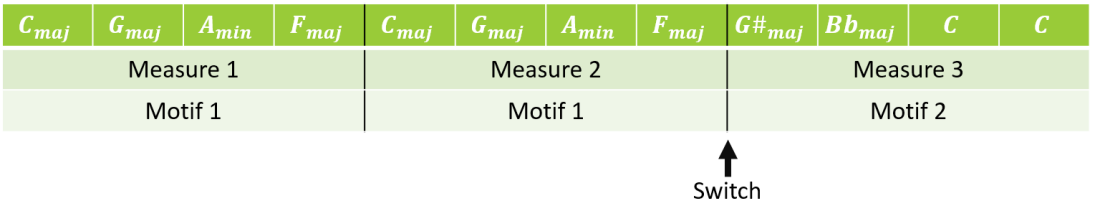


FIGURE 3.9: Representation of the structure of a choreography as a function of the structure of the music.

3.1.4 Description of time and structure

A quadrotor choreography, as a whole, can be mathematically modelled as a switched system. A switched system is a system composed of two different mathematical objects:

- Object 1: differential equations that describe the flow of the continuous dynamics, depending on a continuous control input U.
- Object 2: a switching rule that orchestrates among different subsystems, which is usually called the discrete-event input σ.

Object 1 will refer to the time in a quadrotor choreography. It is the mathematics behind the physical properties of the quadrotor and the control system.

Object 2 is the structure of the choreography. For quadrotors, the structure refers to the changes in path motifs as a function of the changes in music chords. The appropriate sequence of chords can be detected through switching theory. The right sequence of chords could trigger changes in the path motif.

A bar is a structural element of music that consists of a specific number of beats. Figure 3.9 is an example of a simple musical structure consisting of three measures with four bars each. The first two measures have the same chord sequence ($C_{major} - G_{major} - A_{minor} - F_{major}$); they are therefore given the same path motif. The third measure could indicate a change in the structure of the song, such as, for example, a change from a verse to the chorus. The chord sequence of the third measure is different ($G\#_{maj} - Bb_{maj} - C - C$). This change in the chord sequence, detected through the use of a chord-detection algorithm, triggers a change in the current motif. This change is represented by an arrow in figure 3.9.

Note that two different chord sequences that trigger two different path motifs might have the same first chord. For the purposes of this thesis, it is assumed that all chord sequences have different first chords. Since this is not always the case, an alternative is proposed with the use of the MIDI pedals in section 3.4.

3.2 Mathematical formulation of space

This section describes different parametrized path motifs to represent the space the dancing quadrotor will utilize. In all the motif figures in this section, the path motif is the red line, and the red points are the waypoints for integer pitches.

The following musical and physical parameters are used in this section:

- P is the pitch ($P \in [0; 127]$)
- N is the highest pitch an instrument can play
- Q is the maximum number of octaves an instrument can reach ($0 \leq O \leq Q$)
- 12 is the number of semitones in an octave

The physical constraints of the flight space are also needed in this section:

- the allowable maximum flight height is denoted as h_{max} (in meters)
- the minimum flight height is denoted by h_{min} (in meters)

In this section, $h_{max} = 5m$, $h_{min} = 0.3m$, $N = 127$, and $Q = 11$. This work considers three path motifs: the straight line, the circle, and the helix.

3.2.1 Straight line path motif

In this music-to-motif mapping, the concept of "higher pitch" is applied. The path motif is a vertical line, and the pitch of the current note being played by the musician is associated to a particular height. As the melody goes higher in pitch, the quadrotor will hover higher, and conversely the quadrotor will hover lower if the pitch goes lower. The equation of a straight line passing through the point (x_0, y_0, z_0) and parallel to the vector (a, b, c) is characterized by the parametric equations:

$$\begin{aligned} x &= x_0 + a\lambda \\ y &= y_0 + b\lambda \\ z &= z_0 + c\lambda \end{aligned} \tag{3.2}$$

where a, b, c, and λ are real numbers (with $\lambda \neq 0$). Equation (3.2) is most useful when studying the displacement of a body from the starting point (x_0, y_0, z_0) in the (a, b, c)

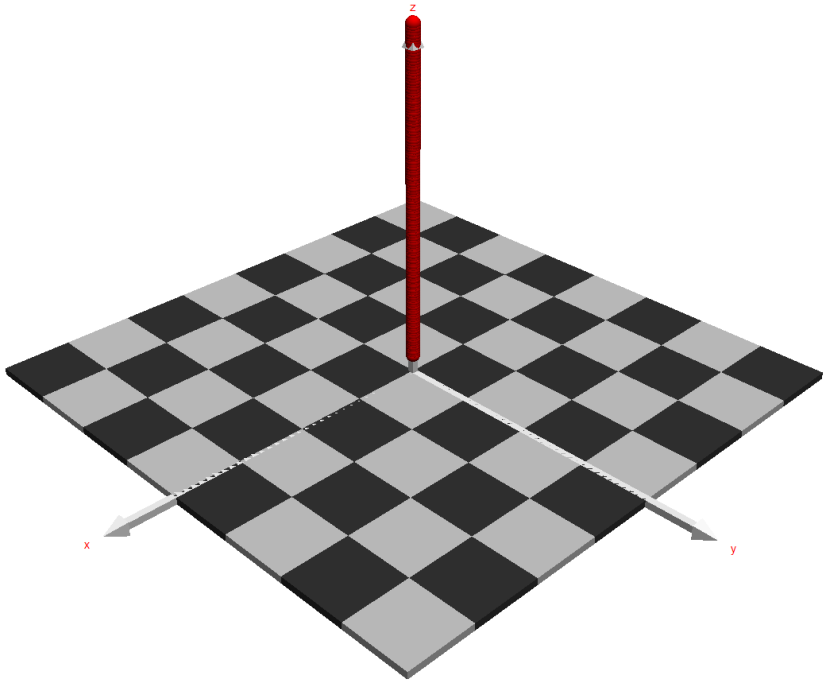


FIGURE 3.10: Simulation of the straight line path motif.

direction. Assume $a = 0$ and $b = 0$ such that $x = x_0$ and $y = y_0$. The quadrotor can then be controlled along the z-axis by varying the λ parameter. By replacing z_0, c and λ by h_{min}, $(h_{max} - h_{min})/N$, and P, respectively, in equation (3.2), equation (3.3) becomes the equation for z. Figure 3.10 is an image from the simulation of the straight line path motif.

$$z = (h_{max} - h_{min})\frac{P}{N} + h_{min} \tag{3.3}$$

3.2.2 Circular path motif

This path motif is an extrapolation of the circular pitch class space (Figure 2.5, see also [80]). A circular trajectory parallel to the $x - y$ plane is parameterized by

$$\begin{aligned} x &= R\cos(\theta) \\ y &= R\sin(\theta) \\ z &= z_0 \end{aligned} \tag{3.4}$$

where x and y are the Cartesian coordinates of the world frame, R is the radius of the circle, θ is the heading angle, and z_0 is the height. A proposed mapping assumes that a zero heading angle corresponds to the note C according to figure 2.1. The trajectories are parameterized by the fixed radius R and by the heading angle defined by:

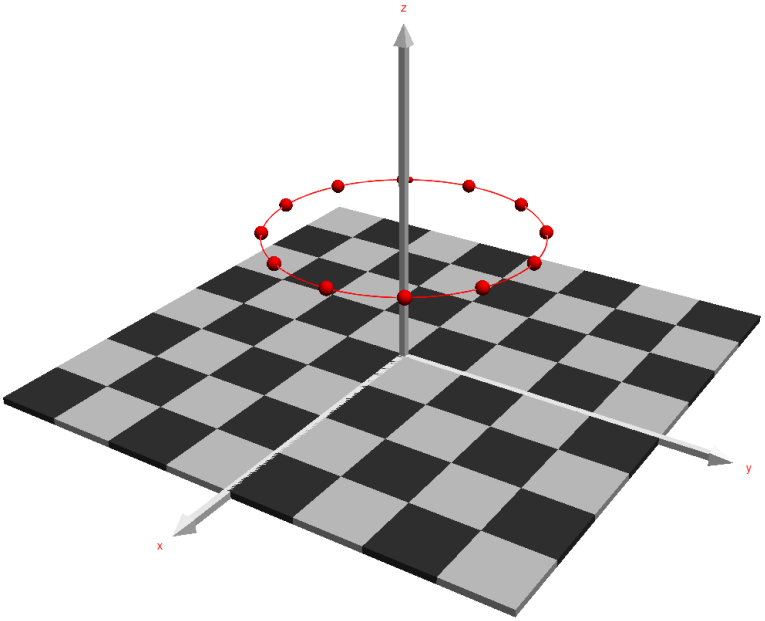


FIGURE 3.11: Simulation of the circle path motif.

$$\theta = 2\pi P/12 \tag{3.5}$$

Figure 3.11 is an image from the simulation of the circle path motif, with $R = 2m$, and $z_0 = 2m$.

3.2.3 Helicoidal trajectory and variations motifs

A helix is parametrized in space by the following equations:

$$\begin{aligned} x &= R\cos(\beta\lambda) \\ y &= R\sin(\beta\lambda) \\ z &= \lambda\alpha + c \end{aligned} \tag{3.6}$$

where R is the radius of the helix, β is the rate of rotation around the z-axis as a function of λ, α is the height rate of climb as a function of λ, z_0 is the base height, and λ is a free parameter. In the first musical mapping, the concept of "higher pitch" and the circle of notes are brought together. Different circles of notes are generated at different heights: the heading angle depends on the pitch, and the height depends on the octave in which the pitch lies. The desired motif is achieved by replacing λ by P, β by $2\pi/12$, α by $(h_{max} - h_{min})/Q$ and c by h_{min}. This path motif is represented in figure 3.12, with $R = 2m$, and can be described by the following equations:

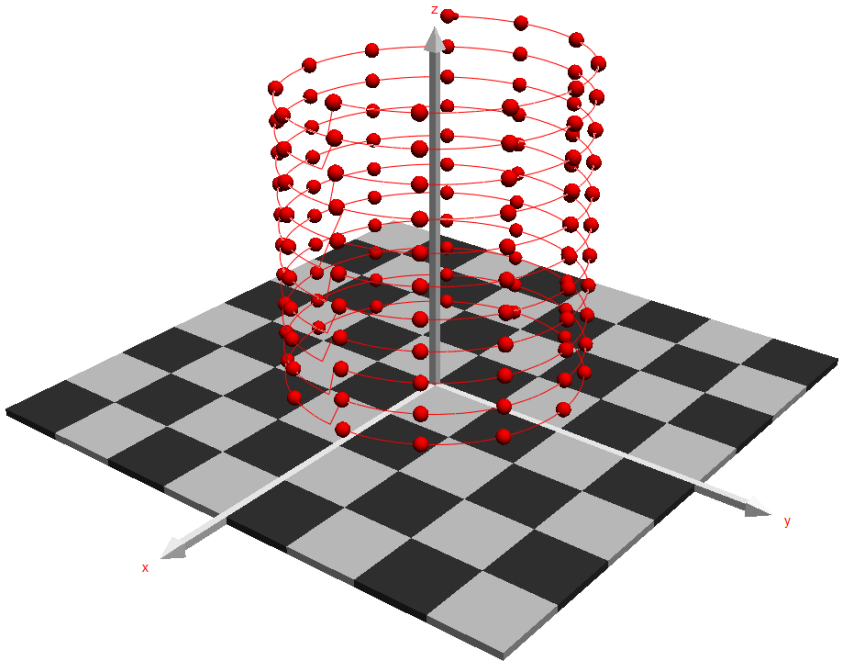


FIGURE 3.12: Simulation of the first helicoidal path motif, with $R = 2m$.

$$\begin{aligned} x &= R\cos(2\pi P/12) \\ y &= R\sin(2\pi P/12) \\ z &= (h_{max} - h_{min})\frac{(P - P \mod (12))}{12Q} + h_{min} \end{aligned} \tag{3.7}$$

This particular motif is discontinuous: the jumps in height are due to the jump in octaves. In a second mapping, the different circles of notes are connected in a more smooth way. The height is continuous and dependent on the pitch instead of the octave. The idea is presented in figure 3.13 and is described by the following equations:

$$\begin{aligned} x &= R\cos(2\pi P/12) \\ y &= R\sin(2\pi P/12) \\ z &= (h_{max} - h_{min})\frac{P}{N} + h_{min} \end{aligned} \tag{3.8}$$

In a third mapping, the circles of notes are smoothly connected since the height is a function of the pitch instead of the octave. However, the radius of the circles is not constant. The radius R is made to decrease as the pitch increases to create a set of circles that form a cone-shaped outline. This can be seen in figure 3.14 and is described by the following equations, where α is fixed:

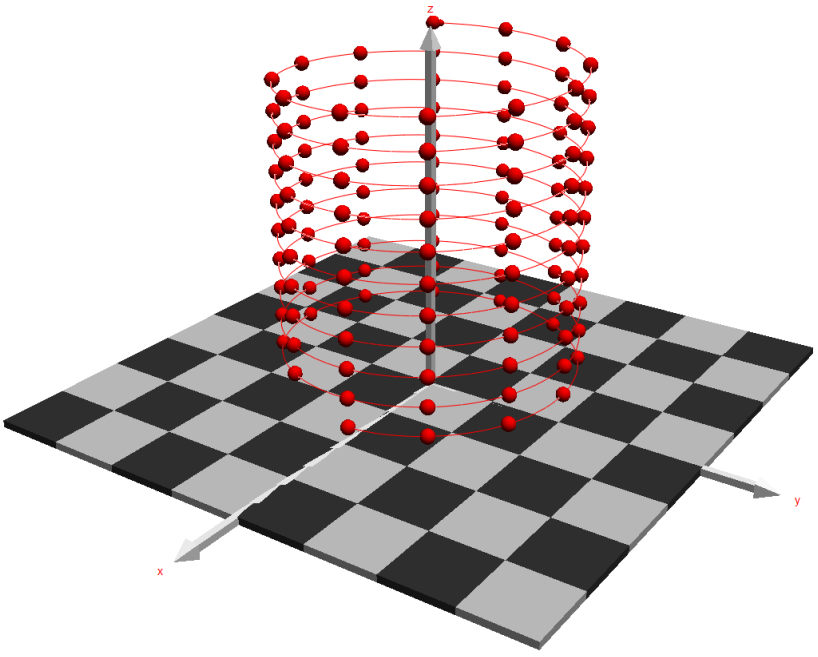


FIGURE 3.13: Simulation of the second helicoidal path motif, with $R = 2m$.

$$\begin{aligned} x &= \alpha(P - N)\cos(2\pi P/12) \\ y &= \alpha(P - N)\sin(2\pi P/12) \\ z &= (h_{max} - h_{min})\frac{P}{N} + h_{min} \end{aligned} \tag{3.9}$$

The path motif possibilities are endless, and their design is only limited by the creativity of the artist.

3.3 Mathematical formulation of shape

The path motifs and the extracted music features generate discrete geographic waypoints. In order to generate a continuous trajectory, the pitch is filtered to generate a 'modified pitch' continuous signal. The proposed solution combines the frequency and the decay of a sound and is a second order system that filters the pitch values to generate a continuous signal. The transfer function of the system is presented in equation (3.10) where ω_n is the undamped natural frequency, and ζ is the damping ratio. Figure 3.15 shows the effect of the changes in ζ and ω_n on the step response of the system. A higher ω_n will result in a faster response, and conversely a lower ω_n results in a slower response. The natural frequency ω_n is the oscillation frequency of the undamped response. The damping ratio ζ controls the overshoot of the oscillations - a higher damping ratio will result in a more damped response with smaller overshoot.

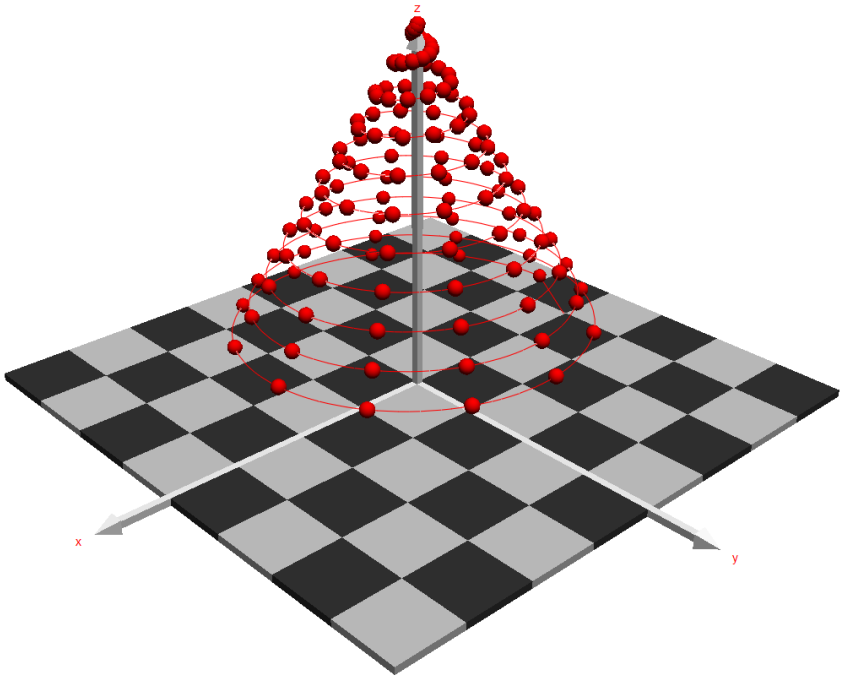
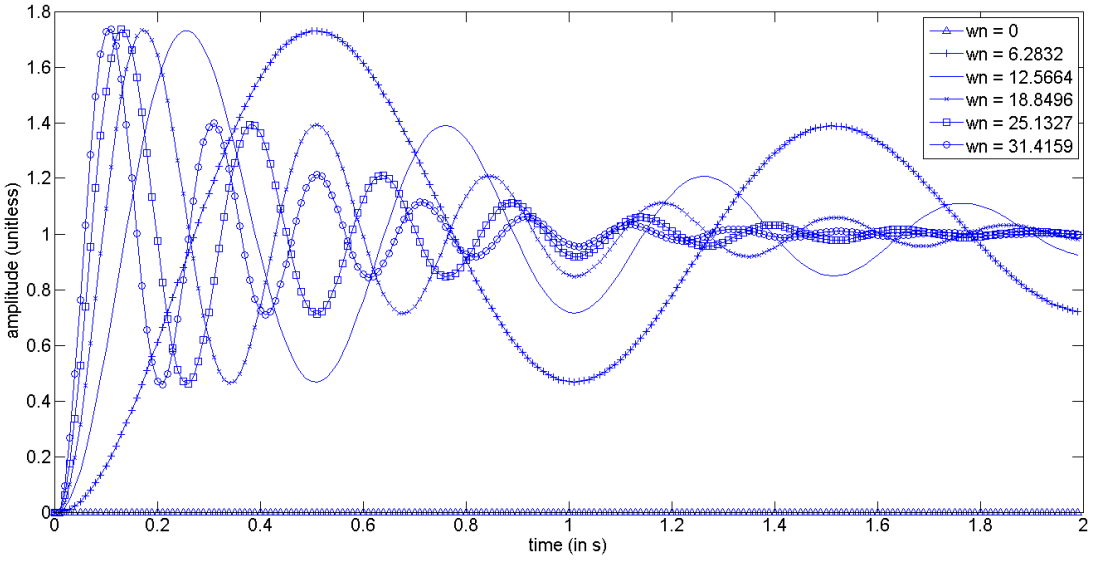


FIGURE 3.14: Simulation of the third helicoidal path motif, with $\alpha = 0.02$.

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n + \omega_n^2} \tag{3.10}$$

As mentioned earlier, the desired motion of the quadrotor is along the path motif, and the pitches generated by the artist are discrete. In order to generate trajectories along the motif, the pitch needs to be continuous. The pitches of the melodic phrase played by the musician are therefore filtered to create a smoother pitch flow. The different responses generated by the shape filter will affect the motion of the quadrotor along the path motif. These different responses can be controlled through the damping ratio and the natural frequency of the second order system. Figure 3.16 is a simulation of the desired motion of the quadrotor along the cone path motif, using the shape filter. Two motions are presented. In both examples, the pitch is increased from 1 to 2, and the natural frequency is maintained at 1 rad/s. The waypoint for $P = 1$ is the point $(-2.18, -1.26, 0.337)$, and the point for $P = 2$ is $(-1.25, -2.16, 0.374)$. In motion 1, the damping ratio is 0.8. It is then changed to 0.3. The shape of the trajectory is different for these two motions: notice the oscillations in the undamped response. Leaving the parameters tunable enables artists to shape the trajectory of the quadrotor along the path motif. These two parameters are made available to the artist through two MIDI pedals: each MIDI pedal generates values between 0 and 127, depending on the position of the pedal. These values are linearly mapped to the allowable range of damping ratio ($\zeta \in [\zeta_{min}, \zeta_{max}]$) and the allowable range of natural frequency ($\omega_n \in [\omega_{n,min}, \omega_{n,max}]$),

(a)



(b)

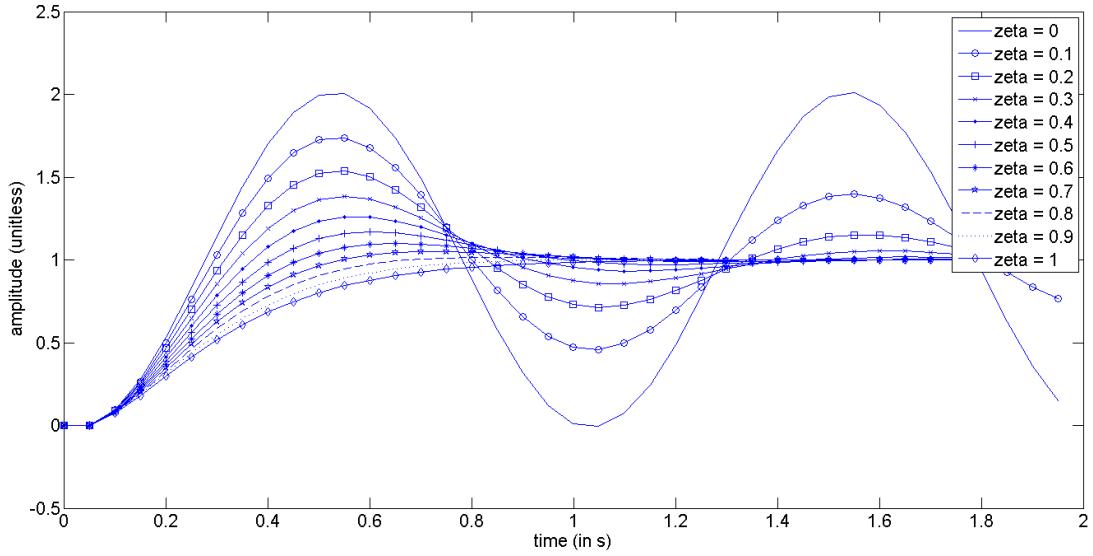


FIGURE 3.15: Second Order system step response with changes in ω_n with $\zeta = 0.1$ (top), and in ζ with $\omega_n = 2\pi$ (bottom).

according to equation (3.11), where E_i is the MIDI value of the i^{th} pedal ($i = 1, 2$). Both parameters are positive.

$$\begin{aligned} \omega_n &= (\omega_{n,max} - \omega_{n,min})\frac{E_1}{127} + \omega_{n,min} \\ \zeta &= (\zeta_{max} - \zeta_{min})\frac{E_2}{127} + \zeta_{min} \end{aligned} \tag{3.11}$$

The code for the shape filter is available in Appendix F.

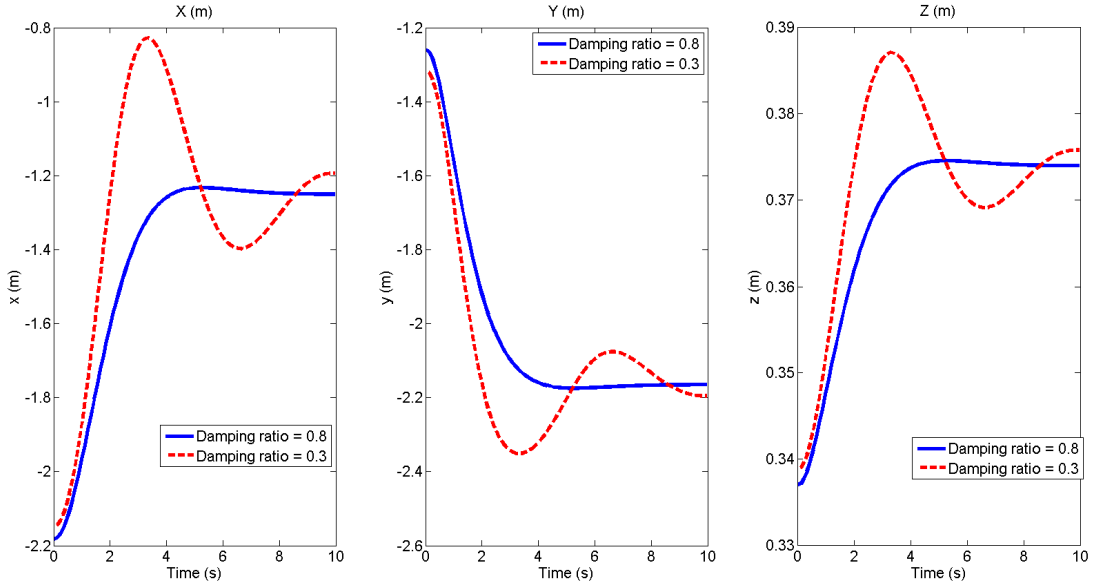


FIGURE 3.16: Desired x, y, and z motions of the quadrotor based on the cone motif using the shape filter.

3.4 Mathematical formulation of structure

The structure of the quadrotor choreography is the changes in path motifs as a function of chord sequences. The chord detection algorithm has been presented in Chapter 2. This section will present the path motif switching algorithm, based on the chords played. Figure 3.18 is a block diagram of the algorithm. The chord sequences are the highlighted blue blocks, and the following variables are used in figure 3.18:

- M is the total number of chord sequences the artist wishes to play, with $M \geq 1$.
- k, m, and n are the total number of chords in chord sequences 1, 2, and N, respectively, with $k, m, n \geq 1$.
- a, b and h refer to specific path motifs.

The example presented in section 3.1 is revisited in figure 3.17 - the table is extended to include information presented in figure 3.18. In the example, there are two different chord sequences ($M = 2$), and each of these chord sequences contains four chords ($k, m = 4$). The path motif will change according to the change from 'chord sequence 1' to 'chord sequence 2'.

A chord sequence has a finite number of chords, and is predefined by the artist. The switching algorithm will follow the predefined chord sequence, such that changes in chord sequences will trigger changes in path motifs. A sequence can be repeated (see figure 3.17). The switching algorithm starts when a chord is detected. If the detected chord is

Measure 1				Measure 2				Measure 3			
C_{maj}	G_{maj}	A_{min}	F_{maj}	C_{maj}	G_{maj}	A_{min}	F_{maj}	$G\#_{maj}$	Bb_{maj}	C_{maj}	C_{maj}
Chord sequence 1				Chord sequence 1				Chord sequence 2			
1.1	1.2	1.3	1.4	1.1	1.2	1.3	1.4	2.1	2.2	2.3	2.4
Path motif 1				Path motif 1				Path motif 2			

Switch

FIGURE 3.17: Popular chord progression with chord follower notation

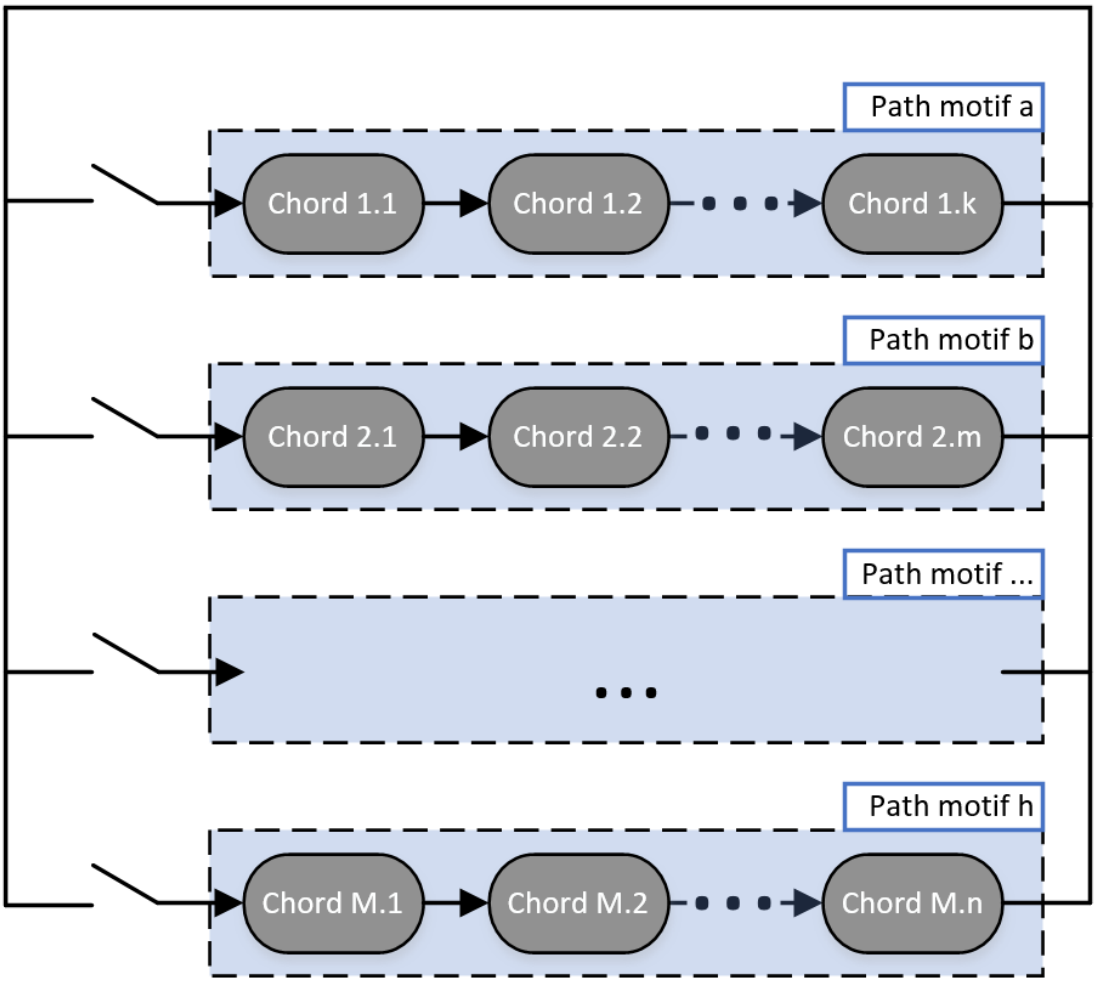


FIGURE 3.18: Block diagram of the structure component

not identical to any of the first chords in the predefined sequences, then the path motif is unchanged. If the detected chord is identical to the first chord in a predefined chord sequence, then the switching algorithm applies the appropriate path motif and waits for a new chord to be played. If the next chord is part of the same chord sequence, then the path motif stays unchanged. The flow chart of the switching algorithm is drawn in figure 3.19.

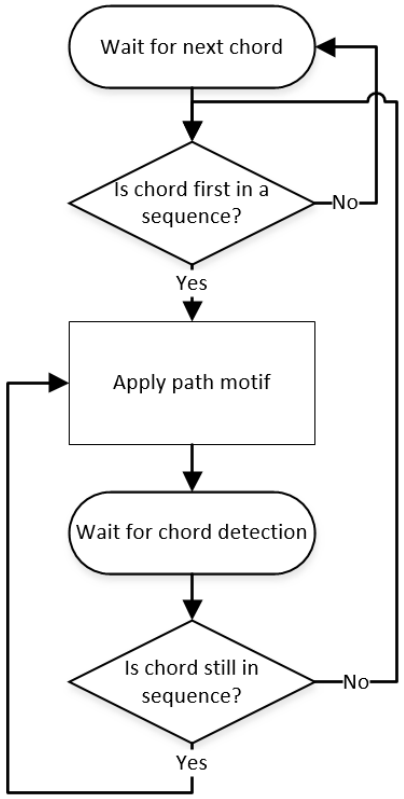


FIGURE 3.19: Switching algorithm, depending on the chords detected

This algorithm is not flexible: it does not allow for musical flexibility throughout a performance, two sequences cannot have the same first chord, and it does not apply to monophonic instruments incapable of playing chords (such as the clarinet or the flute). Therefore, an alternative method is presented. The structure is controlled through MIDI button pedals that trigger changes in path motifs (see figure 3.7). Each path motif is assigned to a button pedal.

3.5 Mathematical formulation of time

In the context of the quadrotor, the timing will refer to the control of the time evolution of the dynamic system. A quadrotor is a Vertical Take-Off and Landing (VTOL) multi-rotor helicopter that is lifted and propelled by four rotors. The quadrotor can move along the x, y and z axes of the inertial reference frame, and rotations are described by ϕ for roll, θ for pitch and ψ for yaw. Figure 3.20 shows a 3-D render of the quadrotor in simulation with the body fixed reference frame attached to it.

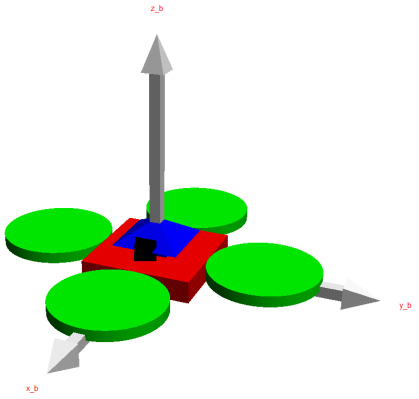


FIGURE 3.20: Image of the simulation of the quadrotor.

3.5.1 Mathematical Model of the Quadrotor

This section presents the nonlinear and the linearized mathematical model of the quadrotor using Newton-Euler's equations.

3.5.1.1 Nonlinear Model

Three assumptions have been made to derive the dynamics of the quadrotor.

1. The quadrotor is a rigid body.
2. The quadrotor rotates around its center of mass, which coincides with the origin of the body fixed reference frame.
3. The quadrotor is symmetrical in the x-z and y-z planes, causing the products of inertia I_{xy}, I_{yz} and I_{xz} to be 0.

The translational dynamics in the inertial reference frame can be described by (see [109], [110]):

$$\begin{aligned} \ddot{x} &= (\sin(\psi)\sin(\phi) + \cos(\psi)\sin(\theta)\cos(\phi))\frac{U_1}{m} \\ \ddot{y} &= (\sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi))\frac{U_1}{m} \\ \ddot{z} &= -g + (\cos(\theta)\cos(\phi))\frac{U_1}{m} \end{aligned} \tag{3.12}$$

where m is the mass (kg) of the quadrotor and g is the magnitude of gravitational acceleration on Earth ($9.81 m/s^2$).

Following a similar approach to [100][109][112], the nonlinear attitude dynamics are described by

$$\begin{aligned}
\ddot{\phi} &= \frac{I_{yy} - I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} - \frac{J_{TP}}{I_{xx}}\dot{\theta}\Omega + \frac{U_2}{I_{xx}} \\
\ddot{\theta} &= \frac{I_{zz} - I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} + \frac{J_{TP}}{I_{yy}}\dot{\phi}\Omega + \frac{U_3}{I_{yy}} \\
\ddot{\psi} &= \frac{I_{xx} - I_{yy}}{I_{zz}}\dot{\phi}\dot{\theta} + \frac{U_4}{I_{zz}} \\
\Omega &= -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4
\end{aligned} \tag{3.13}$$

where Ω_j is the angular speed of the j^{th} motor and J_{TP} is the rotational moment of inertia about the propeller axis.

The control inputs U_i are defined as:

$$\begin{aligned}
U_1 &= T_1 + T_2 + T_3 + T_4 \\
U_2 &= l(-T_1 - T_2 + T_3 + T_4) \\
U_3 &= l(+T_1 - T_2 + T_3 - T_4) \\
U_4 &= k_\psi(-T_1 + T_2 + T_3 - T_4)
\end{aligned} \tag{3.14}$$

where T_i represents the thrust generated by the corresponding i^{th} motor, k_ψ is the drag factor due to yaw, and l is the length in meters from the center of rotation of the motor to the center of mass of the quadrotor. Equation (3.14) represents the vertical, roll, pitch, and yaw control inputs, respectively.

3.5.1.2 Linearized Model

It is assumed that the quadrotor will operate around a hovering state with small angle deflections. Linearizing equation (3.12) about the hovering state, where $\phi \approx 0$, $\theta \approx 0$, and $\psi \approx \psi_0$, results in

$$\begin{aligned}
\ddot{x} &= (cos(\psi_0)\theta + sin(\psi_0)\phi)\frac{U_1}{m} \\
\ddot{y} &= (sin(\psi_0)\theta - cos(\psi_0)\phi)\frac{U_1}{m} \\
\ddot{z} &= -g + \frac{U_1}{m}
\end{aligned} \tag{3.15}$$

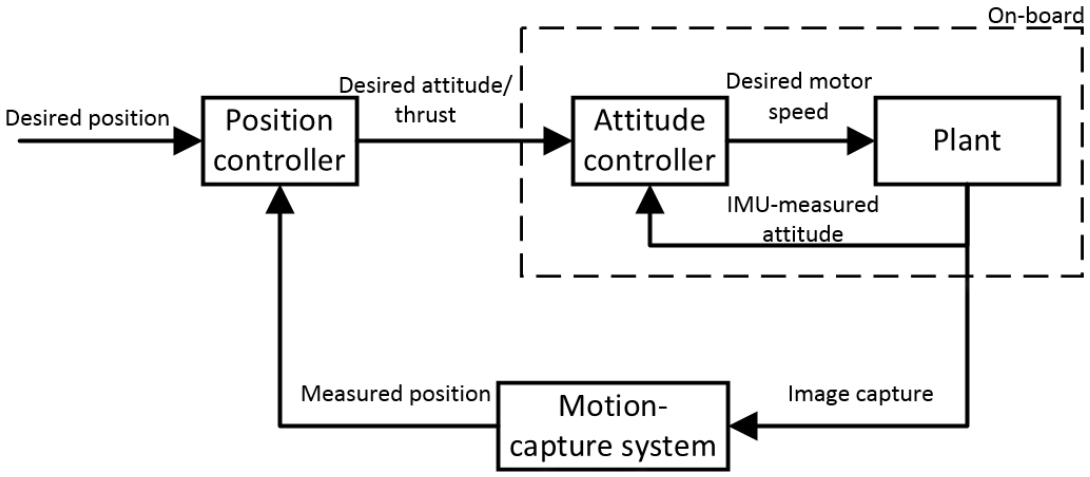


FIGURE 3.21: Controller block diagram in a hardware implementation.

Linearizing equation (3.13) about the hovering state, with $\dot{\phi} \approx 0$, and $\dot{\theta} \approx 0$ and $\dot{\psi} \approx 0$ yields the following linear attitude dynamics

$$\begin{aligned} \ddot{\phi} &= \frac{U_2}{I_{xx}} \\ \ddot{\theta} &= \frac{U_3}{I_{yy}} \\ \ddot{\psi} &= \frac{U_4}{I_{zz}} \end{aligned} \tag{3.16}$$

The model parameters are based on an Arduino prototype, designed in the Hycons lab:

Parameter	Value
$m(kg)$	0.431
$I_{xx}(kg.m^2)$	0.0021
$I_{yy}(kg.m^2)$	0.0018
$I_{zz}(kg.m^2)$	0.0027
$l(m)$	0.19
k_ψ	0.1

3.5.2 Controller Design

The control structure consists of two separate controllers: the position controller and the attitude controller. These two controllers provide a nested closed loop solution, where the position control is the outer loop, and the attitude control is the inner loop (see figure 3.21).

3.5.2.1 Position Controller

The position controller is designed such that the closed loop dynamics are described by equations (3.17), where ω_i is the natural frequency and ζ_i is the damping ratio, for $i = x, y, z$.

$$\begin{aligned} 0 &= \omega_x^2(x_{desired} - x) + 2\zeta_x\omega_x(\dot{x}_{desired} - \dot{x}) + (\ddot{x}_{desired} - \ddot{x}) \\ 0 &= \omega_y^2(y_{desired} - y) + 2\zeta_y\omega_y(\dot{y}_{desired} - \dot{y}) + (\ddot{y}_{desired} - \ddot{y}) \\ 0 &= \omega_z^2(z_{desired} - z) + 2\zeta_z\omega_z(\dot{z}_{desired} - \dot{z}) + (\ddot{z}_{desired} - \ddot{z}) \end{aligned} \tag{3.17}$$

Replacing $\ddot{z}$ from equation (3.15) in equation (3.17) yields the control law for U_1:

$$\frac{U_1}{m} = \omega_n^2(z_{desired} - z) + 2\zeta\omega_n(\dot{z}_{desired} - \dot{z}) + \ddot{z}_{desired} + g \tag{3.18}$$

Solving equations (3.15) at hover conditions (U1=mg) for $\ddot{x}$ and $\ddot{y}$ yields equations (3.19). The values of $\ddot{x}$ and $\ddot{y}$ are then obtained from equations (3.17).

$$\begin{aligned} \phi_{desired} &= \frac{1}{g}(\ddot{x}sin(\psi_0) - \ddot{y}cos(\psi_0)) \\ \theta_{desired} &= \frac{1}{g}(\ddot{x}cos(\psi_0) + \ddot{y}sin(\psi_0)) \end{aligned} \tag{3.19}$$

The values of U_1, $\phi_{desired}$ and $\theta_{desired}$ are then processed by the attitude controller.

3.5.2.2 Attitude Controller

The attitude controller is responsible for the orientation of the quadrotor. It receives the desired roll, pitch, and yaw angles from the position controller and computes the control inputs U_2, U_3, and U_4. The linearized roll, pitch, and yaw dynamics are similar to each other, and so are the mathematics behind the controllers. Therefore, only the roll controller is presented in detail. The following notation is used:

- ϕ, $\dot{\phi}$ and $\ddot{\phi}$ are the roll, the roll rate and the roll acceleration, respectively. In a hardware implementation, these values are read and calculated from the on-board inertial measurement unit.
- $\phi_{desired}$, $\dot{\phi}_{desired}$ and $\ddot{\phi}_{desired}$ are the desired roll, the desired roll rate and the desired roll acceleration, respectively. The desired roll is obtained from equation

(3.19), while the desired roll rate and acceleration are calculated from the changes in the desired angle.

The linearized system for roll is described by the following equation:

$$\ddot{\phi} = \frac{U_2}{I_{xx}} \tag{3.20}$$

The control input U_2 can be designed such that the dynamics for roll behave like a second order system with damping ratio ζ and natural frequency ω as follows:

$$0 = \omega^2(\phi_{desired} - \phi) + 2\zeta\omega(\dot{\phi}_{desired} - \dot{\phi}) + (\ddot{\phi}_{desired} - \ddot{\phi}) \tag{3.21}$$

By including the dynamics for $\ddot{\phi}$ from equation (3.20) in the desired second-order response, we get the following control law for U_2:

$$\frac{U_2}{I_{xx}} = \omega^2(\phi_{desired} - \phi) + 2\zeta\omega(\dot{\phi}_{desired} - \dot{\phi}) + \ddot{\phi}_{desired} \tag{3.22}$$

It is noteworthy to compare the simulation implementation to a hardware implementation. In the simulation, the values of the desired and actual angular rates and accelerations $\dot{\phi}_{desired}$, $\ddot{\phi}_{desired}$, $\dot{\phi}$ and $\ddot{\phi}$ are numerically derived. In a hardware implementation, the numerical derivation might lead to a noisy signal, and an unstable controller. Therefore, for a physical implementation, a proportional feedback law for U_2 is preferred, where k is a real number:

$$U_2 = k(\phi - \phi_{desired}) \tag{3.23}$$

The pitch and yaw controllers are designed following the same procedure.

3.5.3 Controller performance

In this section, the step response of the simulated system is measured. The quadrotor is first commanded fixed values of orientation angles, and then position coordinates.

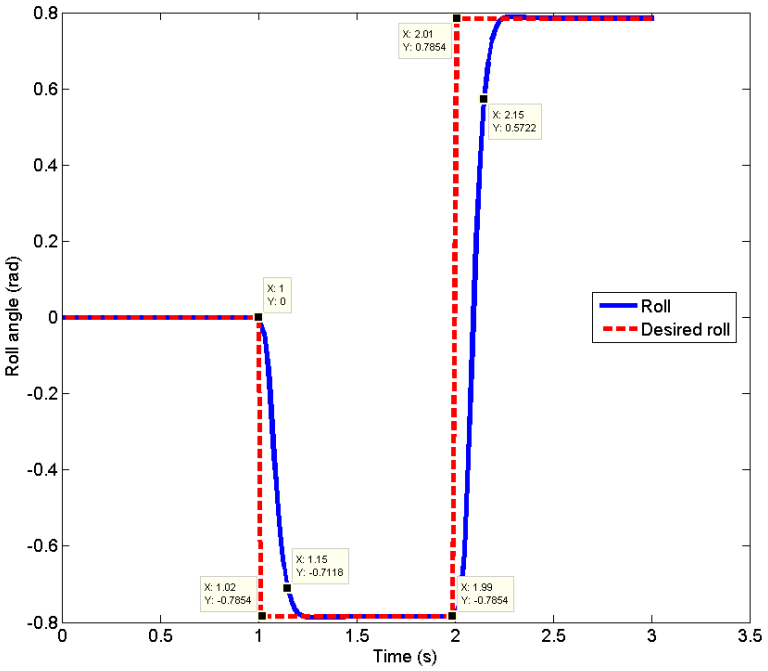


FIGURE 3.22: Response of the attitude controller for roll

3.5.3.1 Attitude controller performance

Since the linear dynamics for roll, pitch, and yaw are identical, the controller for the different orientations are designed similarly. Therefore, only the roll controller is presented. The attitude controller natural frequency and damping ratio are:

- $\omega_n = 2\pi$
- $\zeta = 0.6$

The quadrotor starts with a roll angle of 0. At $t = 1$, the quadrotor is assigned a roll angle of $-\frac{\pi}{4} rad$, and a roll angle of $\frac{\pi}{4} rad$ at $t = 2$. The rise time is $t_r = 0.15s$, with no overshoot.

3.5.3.2 Position controller performance

The position controller is divided into a height controller and a position reference angle generator. The height controller is presented first. The height controller natural frequency and damping ratio are:

- $\omega_n = 3\pi$

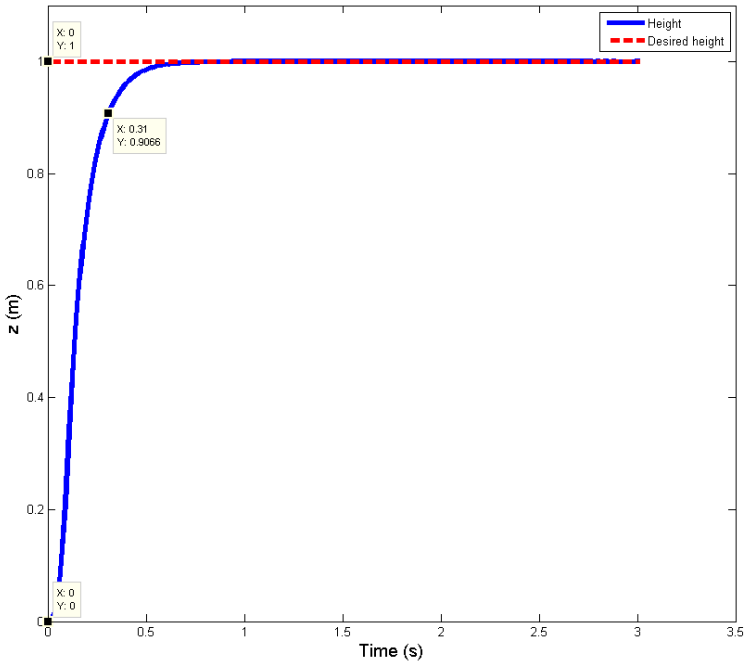


FIGURE 3.23: Height of the quadrotor system for a motion in the z-axis

- $\zeta = 0.6$

At $t = 0$, the quadrotor is assigned a height of $z = 1m$. Figure 3.23 shows the response of the quadrotor from rest at $z = 0m$. The rise time is $t_r = 0.3s$, with no overshoot.

As a reminder, the horizontal position controller consists of a reference angle generator sending the desired angles to the attitude controller (see figure 3.21). The angle generator natural frequency and damping ratio are:

- $\omega_n = 2\pi$
- $\zeta = 1.2$

Figure 3.24 is a response of the quadrotor for a displacement from $x = 0m$ to $x = 1m$. The rise time is $t_r = 0.76s$, and the system overshoots by 13%. The settle time is $t_s = 1.3s$.

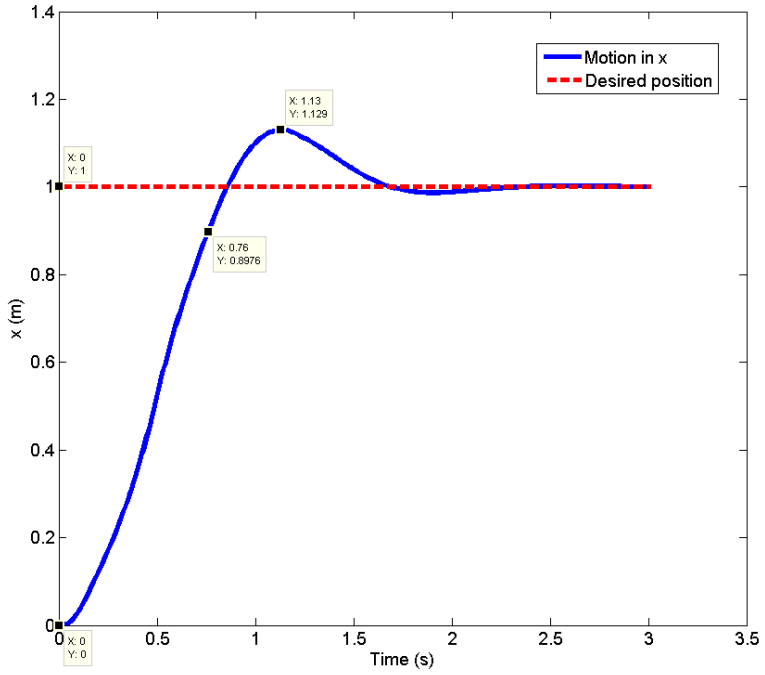


FIGURE 3.24: Response of the quadrotor system for a motion in the x-axis

Chapter 4

Simulation and validation

The mathematical formulation of a choreography was presented in Chapter 3. The contribution of this chapter is to develop a new simulation tool, programmed in Python, to validate the quadrotor choreography. Section 4.1 presents the details of the hardware-in-the-loop implementation of the quadrotor choreography simulation. In section 4.2, the position of the quadrotor is used to play back music to the musician. Section 4.3 presents a validation of the functionality of the system. The complete code of the quadrotor simulation is available in Appendices D to J.

4.1 Hardware-in-the-loop simulation

This section will present the software implementation of the quadrotor choreography, along with all the hardware used. Figure 4.1 is an image of a hardware-in-the-loop simulation of the quadrotor choreography system, where the laptop, the pedal board, and the MIDI keyboard are clearly visible. The hardware is presented first. The software will be presented second.

4.1.1 Hardware

There are three hardware components in the simulation implementation. The following is a list of the components, visible in figure 4.1.

- The simulation is run on a Samsung Series 9 laptop running Windows 10 Technical Preview. The laptop has 8GB of RAM and an Intel i5 1.7GHz CPU. However, there are no restrictions to the operating system: the simulation is written in Python, and the MIDI standard is available on all operating systems.

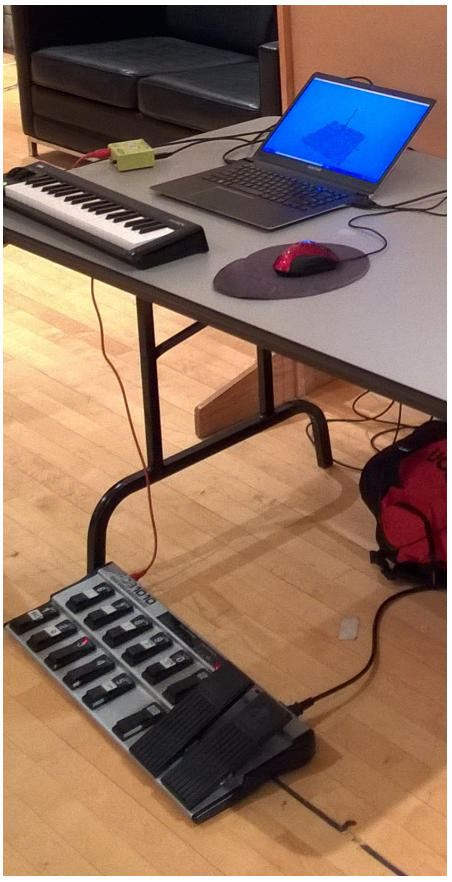


FIGURE 4.1: Image of quadrotor simulation at the CIRMMT symposium booth

- A MIDI keyboard is used to play music to the quadrotor. The korg microKEY-37 MIDI keyboard connects to the main computer through a USB cable [115]. Using a digital musical instrument focuses the scope of the thesis on the implementation of the mathematical formulation of a choreography. An acoustic musical instrument and a pitch detection algorithm could also be used, as discussed in Chapter 2.

- The Behringer FCB1010 [93] MIDI pedal board has two expressive pedals to control the damping ratio and the natural frequency of the shape filter, and 10 button pedals to control the path motif (see Chapter 3). The physical output on the pedal is a MIDI connector - a converter is required to connect the pedal to a computer through USB (see figure 4.1).

4.1.2 Software

The system is simulated in a Python environment. "Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very

attractive for Rapid Application Development" [116]. Two open-source Python add-ons have been used:

- VPython is a graphical library which gives programmers the tools to draw 3-D objects [117].
- The Pygame library is a set of modules designed for programming video games [118]. One of the modules reads MIDI hardware input events and plays MIDI music. [119].

For code clarity, the simulation code has been divided into different files.

- The *main* file contains the linearized equations of translational and rotational dynamics. *main* is also responsible for drawing the simulation, and playing the music generated by the quadrotor back to the musician. This file also starts the threads for the controllers, the shape filter, and the TCP listener. This code is available in Appendix D.
- *controller* contains the position and attitude controllers of the quadrotor. The file consists of a function which receives the current and desired position, velocity, and acceleration of the quadrotor, and returns the control inputs $U_{1,\ldots,4}$. This file contains code that runs on a thread related to the main process. This code is available in Appendix E.
- *shape* is the shape filter for the pitch played to the quadrotor. This file contains code that runs on a thread related to the main process. This code is available in Appendix F.
- *motifs* contains a function that takes two parameters (the motif number and a pitch), and returns the desired trajectory. This file also contains code for the function that retrieves the properties of the music the quadrotor plays back to the musician. This code is available in Appendix G.
- *TCP* receives the MIDI commands from the separate *MIDI* process. This file contains code that runs on a thread related to the main process. This code is available in Appendix H.
- A separate *MIDI* process reads the USB MIDI inputs from the MIDI keyboard and the pedal board. This process also transmits the MIDI information through TCP to the *main* process. This code is available in Appendix I.

- A *model* file contains all the simulation and quadrotor parameters, such as the simulation frequency and the moments of inertia of the quadrotor. This code is available in Appendix J.

The simulation is run at a frequency of $100Hz$. The controllers and the shape filter are transformed to a discrete time equivalent system through the bilinear transform in equation (4.1), where T is the samping period, s is the derivative operator, and z^{-1} is the one step advance operator:

$$s = \frac{2}{T}\frac{z-1}{z+1} \tag{4.1}$$

The bilinear transform discretizes continuous systems. The left half-plane of the Laplace domain is mapped to the unit circle of the z-domain, and therefore the bilinear transform preserves stability [111].

4.2 Quadrotor music playback

The desired position of the quadrotor is along the path motif and, as explained in Chapter 3, the path motif is a one-dimensional manifold embedded in a 3-D space where the musical parameter that controls the desired position of the quadrotor along the path motif is the pitch. Any point along the manifold is assigned at least one pitch. Retrieving the pitch of the path motif from the position of the quadrotor allows for a musical interaction between the quadrotor and the musician. Even though control systems are designed such that the quadrotor follows the desired trajectory, the actual position of the quadrotor might not belong to the path motif. This section presents a music generation algorithm depending on the position of the quadrotor and the path motif.

When generating MIDI music, multiple parameters can be modified to better design the sound. There are two main parameters:

- The pitch of the sound.
- The amplitude of the sound.

The tracking error between the quadrotor's position and the path motif can be characterized by, incidentally, two parameters as well:

- The shortest distance between the path motif and the quadrotor
- The closest point to the quadrotor which belongs to the path motif

The tracking error parameters and MIDI generation parameters are mapped such that:

- The distance between the path motif and the quadrotor is mapped to the amplitude of the sound.
- The pitch of the sound is extracted from the closest point to the quadrotor that belongs to the path motif.

The distance and the pitch played by the quadrotor can be retrieved through analytical or heuristic methods. Both methods use the same notation:

- The pitch associated to the position of the quadrotor is P.
- The total number of pitches, N, is 128.
- h_{max} and h_{min} are the maximum and minimum allowable flight heights, respectively.

The distance between the path motif and the quadrotor is mapped to the amplitude A of the sound generated by the music motif, as seen in equation (4.2) - the amplitude will be lower if the quadrotor is further from the motif, and the amplitude will be 0 if the quadrotor is further than one meter away from the path motif. The MIDI standard is used to play the music generated by the quadrotor - the maximum and minimum amplitudes are 127 and 0, respectively [70].

$$A = \begin{cases} (A_{min} - A_{max})d + A_{max} & d \in [0; 1] \\ 0 & d > 1 \end{cases} \tag{4.2}$$

The analytical solution is presented first in section 4.2.1. Section 4.2.2 presents the heuristic solution.

4.2.1 Analytical solution

Analytically retrieving the pitch from a point along the manifold involves inverting the equations of the path motifs, such that a point along the path motif is mapped to a pitch. The straight line path motif and the circular path motif are presented. To the best of the author's knowledge, there is no analytical solution for the shortest distance between a point and a helix.

4.2.1.1 Straight line path motif

The straight line motif follows the concept of higher pitch - as the melody goes higher in pitch, the quadrotor will hover higher, and conversely the quadrotor will go lower if the pitch goes lower. The pitch can be extracted from the height of the quadrotor through the following equation:

$$P = N\frac{z - h_{min}}{h_{max} - h_{min}} \tag{4.3}$$

The distance between the quadrotor and the z-axis can be calculated from equation 4.4:

$$d = \sqrt{x^2 + y^2} \tag{4.4}$$

4.2.1.2 Circular path motif

The circular path motif is based on the circle of notes. In this mapping, the same note at different octaves is mapped to the same quadrotor waypoint. Therefore, the music note (or the pitch class) being played can be retrieved from the position of the quadrotor, while the pitch of the music (or the octave in which the pitch class lies) cannot. It is assumed that the note being played by the quadrotor lies in the fourth octave. The heading angle of the position of the quadrotor is essential, and is obtained through the following equation:

$$\theta = tan^{-}1(y/x) \tag{4.5}$$

The pitch class of the music motif is then extracted through equation 4.6, and is placed in the fourth octave. As a reminder, there are 12 semi-tones in an octave.

$$P = \frac{12\theta}{2\pi} + 4 \times 12 \tag{4.6}$$

The shortest distance between a circle (parallel to the $x-y$ plane and with center along the z-axis at $z = z_{center}$) and any point is obtained through the following equation:

$$d = \sqrt{(\sqrt{x^2 + y^2} - R)^2 + (z - z_{center})^2} \tag{4.7}$$

This inverse mapping is achieved through the Python math library function, *atan2*. This function takes two parameters (y, x) and returns an angle between $-\pi$ and π, handles the singularities at angles $-\frac{\pi}{2}$ and $\frac{\pi}{2}$, and at the point $(x, y) = (0, 0)$ [120].

4.2.2 Heuristic solution

Since the pitches are discrete, the space is limited to 128 points along the motif. Measuring the distance d between the quadrotor and 128 waypoints is not a computationally expensive operation on computers today. Therefore, the distance d, and the pitch P of the closest point along the path motif to the quadrotor S, are obtained through an extensive search.

The distances between pitch waypoints of the path motif and the quadrotor is measured through equation (4.8) for all pitches between 0 and 127, where (x_S, y_S, z_S) are the coordinates of the quadrotor, and (x_P, y_P, z_P) are the coordinates of a waypoint. Only the shortest distance and the corresponding pitch are used in the music generation process.

$$d = \sqrt{(x_P - x_S)^2 + (y_P - y_S)^2 + (z_P - z_S)^2} \tag{4.8}$$

The following code returns the pitch of the waypoint along the path motif which is closest to the quadrotor, along with the MIDI scaled amplitude.

```
def inverseMotif(motifNumber, x, y, z):
    distance = 100
    closestPitch = 1
    for numericalPitch in my_range(0,127,1):
        numX, numY, numZ = motif(motifNumber, numericalPitch)
        distanceTemp = sqrt ( (x - numX)**2 + (y - numY)**2 + (z - numZ)**2)
        if (distanceTemp < distance):
            distance = distanceTemp
            closestPitch = numericalPitch
    closestPitch = round(closestPitch)
    amplitude = round(-127*distance + 127)
    return closestPitch, amplitude
```

The sound generated from the music motif equations is played back to the musician in real-time through the pygame MIDI library. The timber of the sound, which distinguishes different types of sound production sources, such as voices and musical instruments, string instruments, wind instruments, and percussion instruments, is unaltered throughout the performance. The timber could be associated to a path motif, or can be changed through the Behringer FCB1010 MIDI pedal board. Other music generation parameters can be modified throughout the performance.

4.3 System validation

The complete quadrotor choreography system is validated in this section. The system is tested for different values of music and choreography parameters. The parameters of the mathematical formulation of the choreography, and the music the quadrotor dances to, are changed as a function of time. The path motif switching element will be time dependent. The shape filter parameters, ζ and ω are changed twice for each motif. The pitch is changed once for each variation of the previously presented parameters. Section 4.3.1 will present the straight line motif. Section 4.3.2 contains results of the circle motif. Section 4.3.3 and section 4.3.4 show the helix and the cone motifs, respectively.

Each section contains 4 figures. The first figure is a screen capture of the quadrotor simulation implementation. The path motif is drawn in red. The second figure shows the changes in the shape parameters ζ and ω, along with the distance between the quadrotor and the path motif, as a function of time. The third figure compares the music pitch, the filtered pitch (called shaped pitch), and the pitch the quadrotor plays back to the musician. Notice the changes in the response of the shaped pitch as a function of the variations in the shape parameters ζ and ω. As a reminder, the music played back to the musician by the quadrotor are integer pitches played through the Pygame MIDI library. The fourth figure represents the x, y and z position of the actual and the desired position of the quadrotor, as a function of time.

4.3.1 Straight line path motif

Figure 4.2 is an image of the quadrotor simulation with the straight line motif. Figure 4.3 shows the changes in the shape parameters ζ and ω, along with the changes in the distance between the quadrotor and the path motif. Figure 4.4 displays the changes of the music pitch, the pitch processed by the shape filter, and the pitch the quadrotor plays back to the musician. The position of the quadrotor and of the desired position of the quadrotor are shown in figure 4.5.

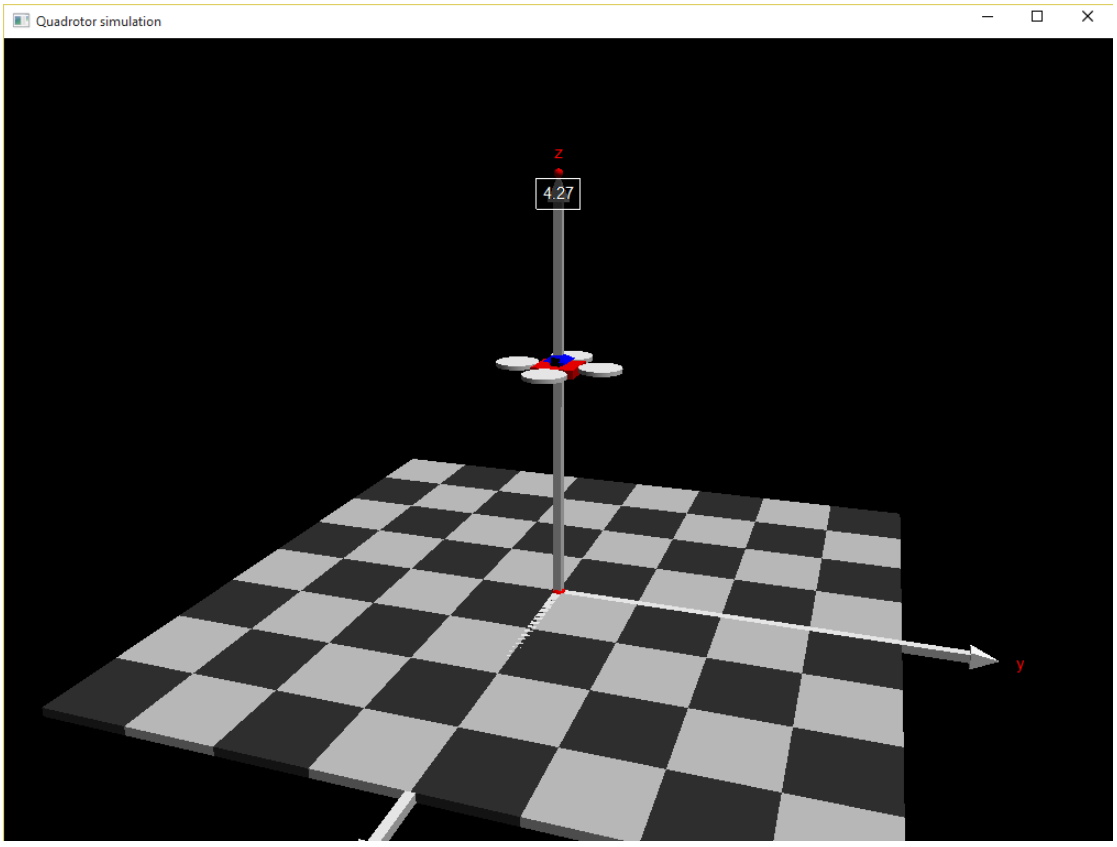


FIGURE 4.2: Screen capture of the quadrotor simulation with the straight line motif

4.3.2 Circle path motif

Figure 4.6 is an image of the quadrotor simulation with the circle path motif. Figure 4.7 shows the changes in the shape parameters ζ and ω, along with the changes in the distance between the quadrotor and the path motif. Figure 4.8 displays the changes of the music pitch, the pitch processed by the shape filter, and the pitch the quadrotor plays back to the musician. Notice how the pitch is contained within the fourth octave ($P \in [48 - 60]$). Since the octave in which the pitch class lies cannot be retrieved, it is assumed that the pitch class lies in the fourth octave. The position of the quadrotor and of the desired position of the quadrotor are shown in figure 4.9.

4.3.3 Helix path motif

Figure 4.10 is an image of the quadrotor simulation with the helix path motif. Figure 4.11 shows the changes in the shape parameters ζ and ω, along with the changes in the distance between the quadrotor and the path motif. Figure 4.12 displays the changes of the music pitch, the pitch processed by the shape filter, and the pitch the quadrotor

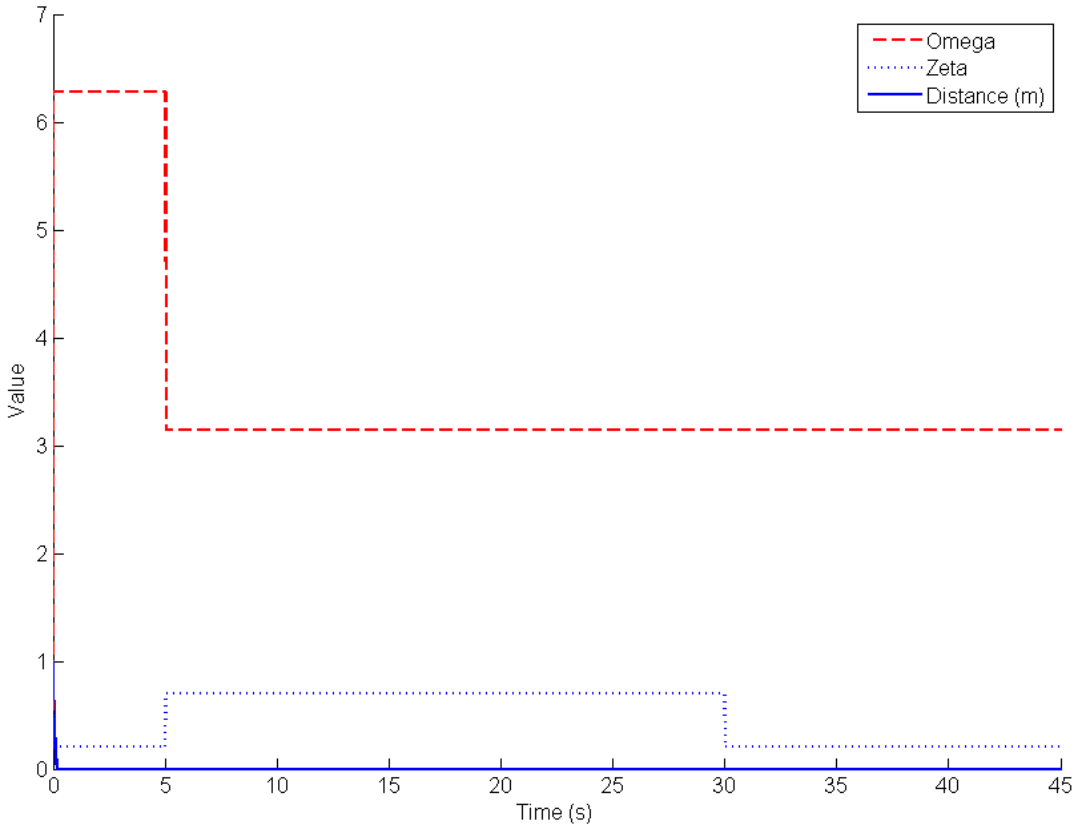


FIGURE 4.3: Changes of the shape parameters (the damping ratio and the natural frequency) and the distance as a function of time.

plays back to the musician. The position of the quadrotor and of the desired position of the quadrotor are shown in figure 4.13.

4.3.4 Cone path motif

Figure 4.10 is an image of the quadrotor simulation with the cone path motif. Figure 4.15 shows the changes in the shape parameters ζ and ω, along with the changes in the distance between the quadrotor and the path motif. Figure 4.16 displays the changes of the music pitch, the pitch processed by the shape filter, and the pitch the quadrotor plays back to the musician. The position of the quadrotor and of the desired position of the quadrotor are shown in figure 4.17.

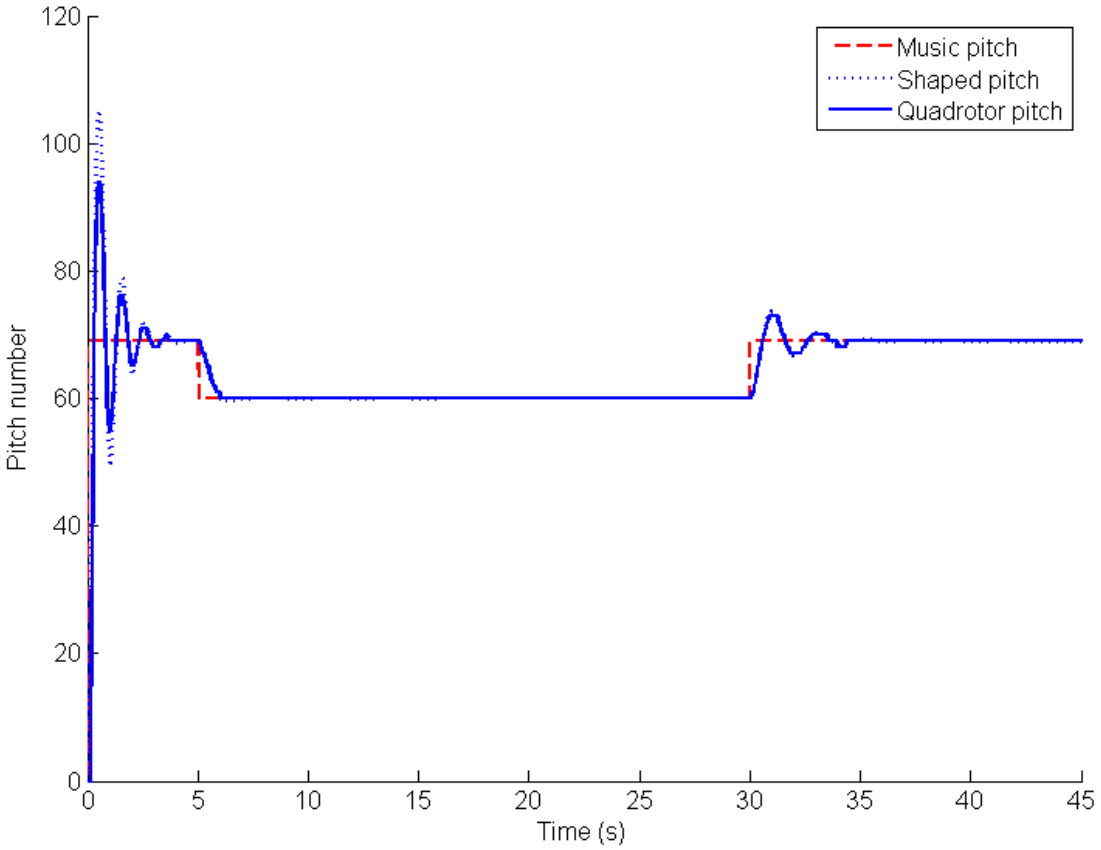


FIGURE 4.4: Changes of the pitch, the shaped pitch and the pitch played by the quadrotor.

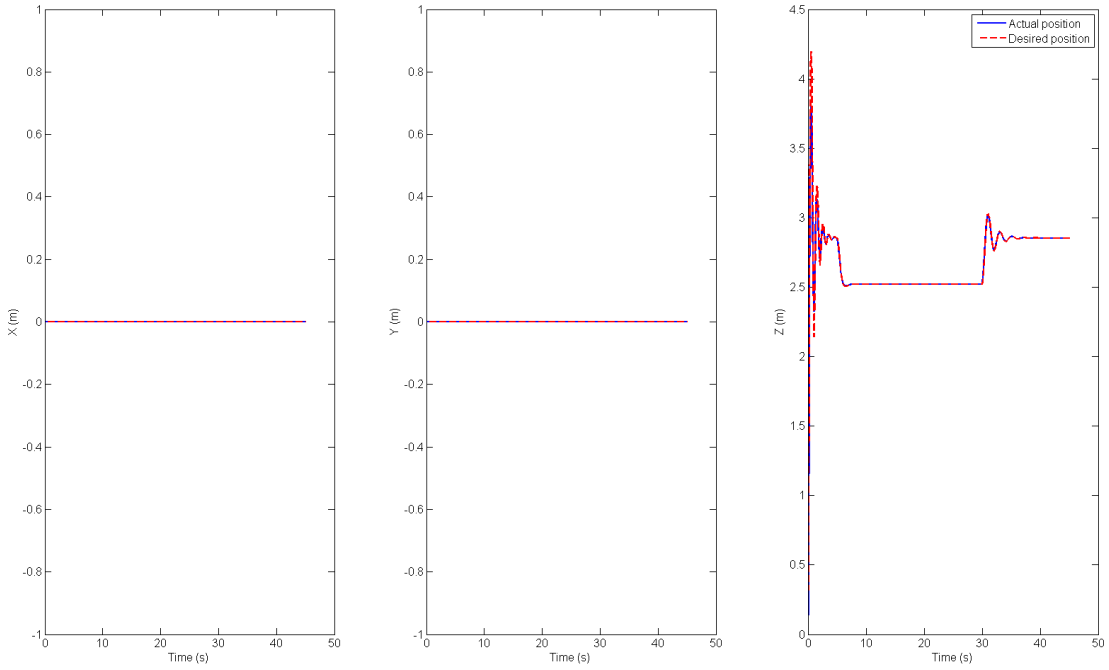


FIGURE 4.5: Position of the quadrotor and the desired trajectory.

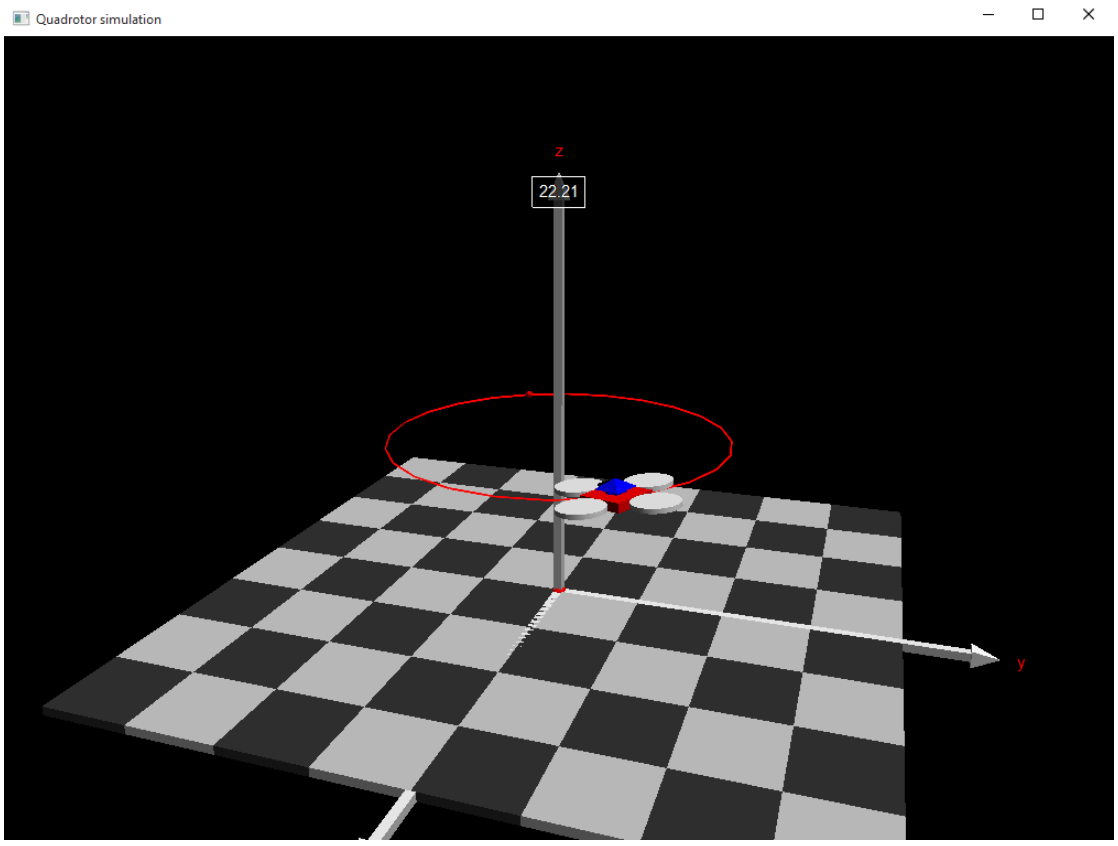


FIGURE 4.6: Screen capture of the quadrotor simulation with the circle motif

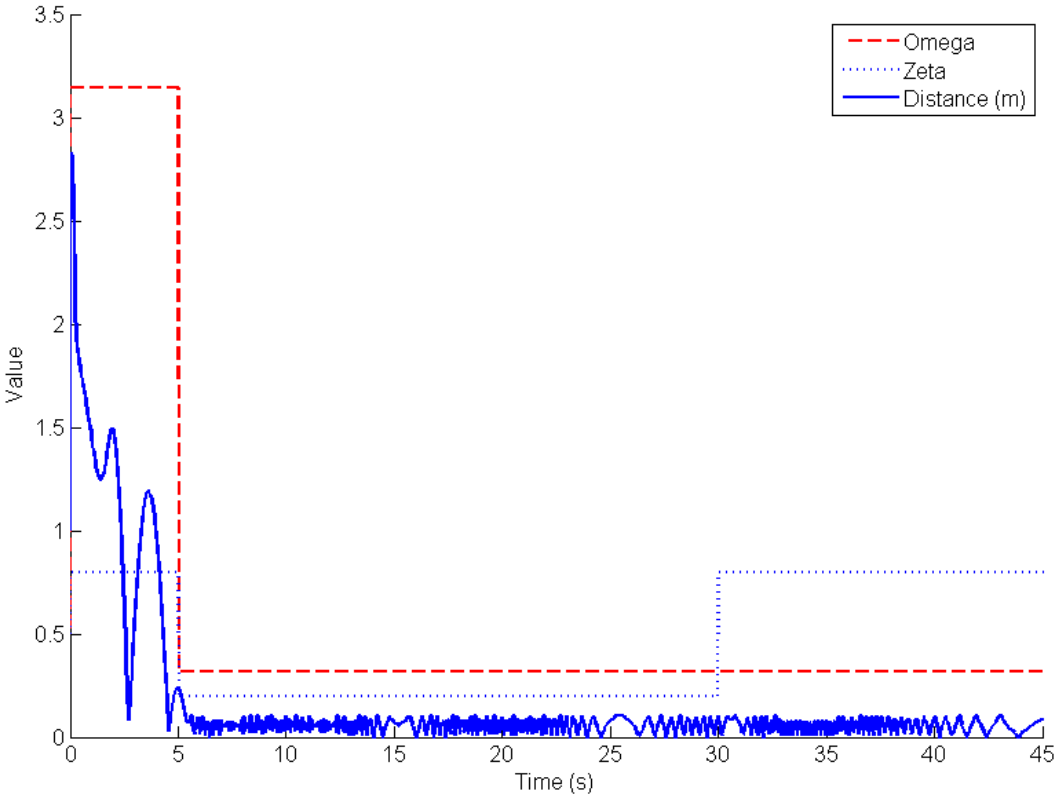


FIGURE 4.7: Changes of the shape parameters (the damping ratio and the natural frequency) and the distance as a function of time.

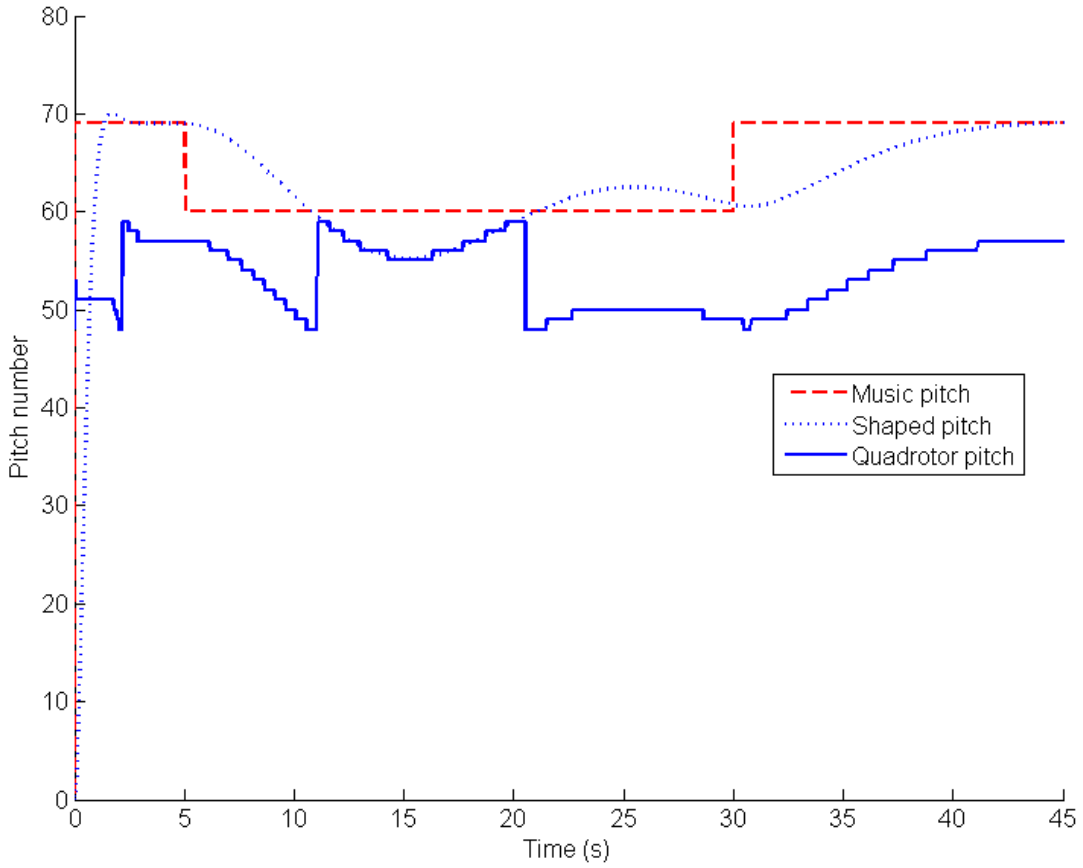


FIGURE 4.8: Changes of the pitch, the shaped pitch and the pitch played by the quadrotor.

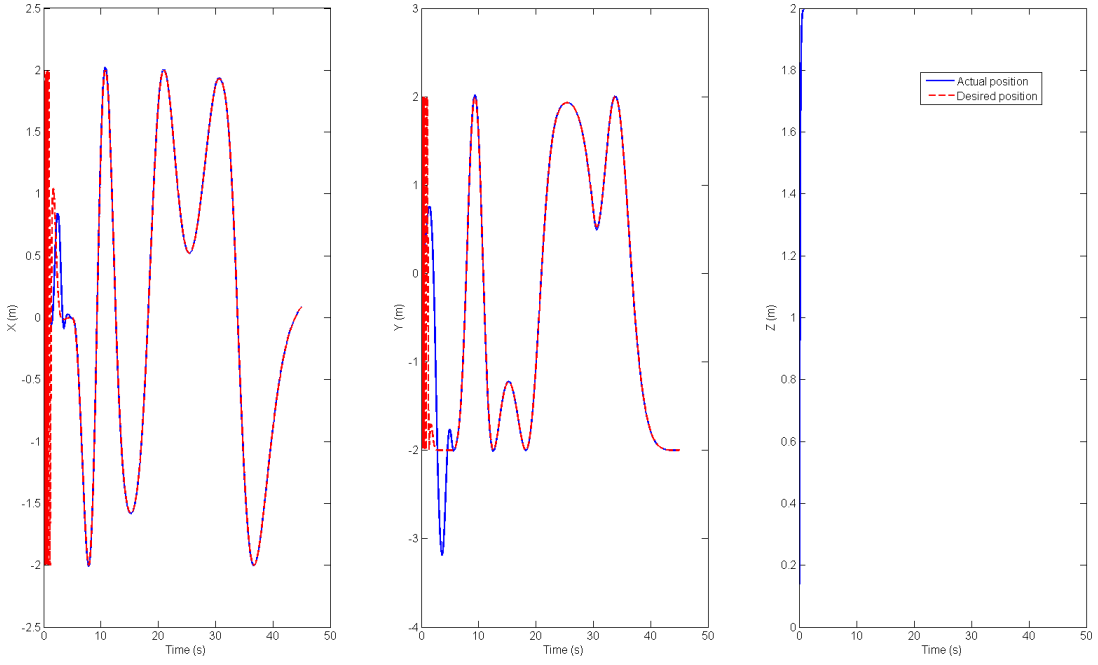


FIGURE 4.9: Position of the quadrotor and the desired trajectory.

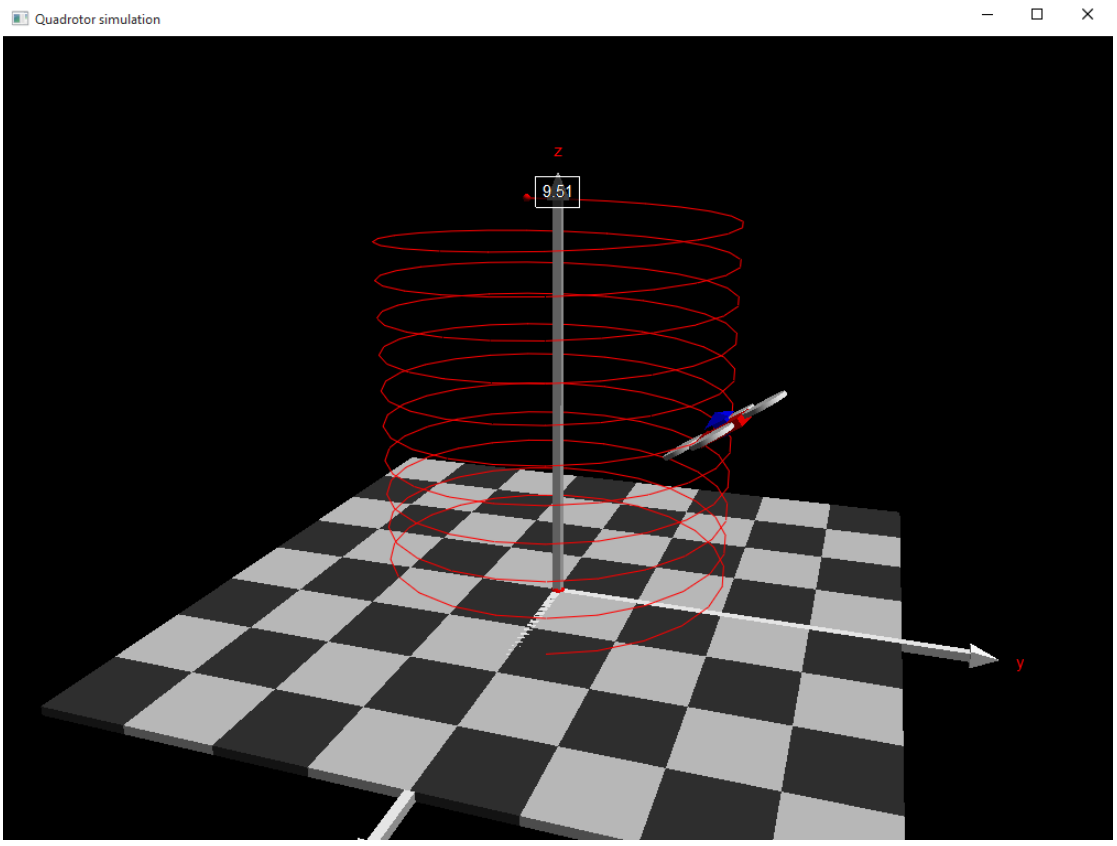


FIGURE 4.10: Screen capture of the quadrotor simulation with the helix motif

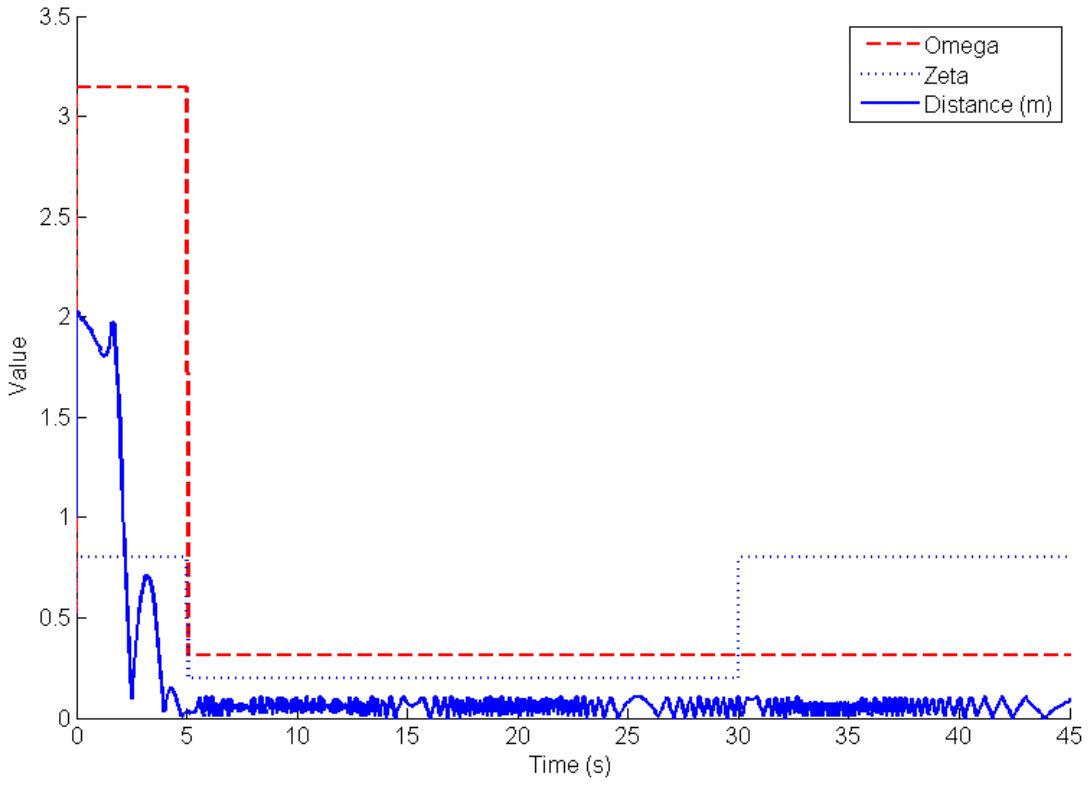


FIGURE 4.11: Changes of the shape parameters (the damping ratio and the natural frequency) and the distance as a function of time.

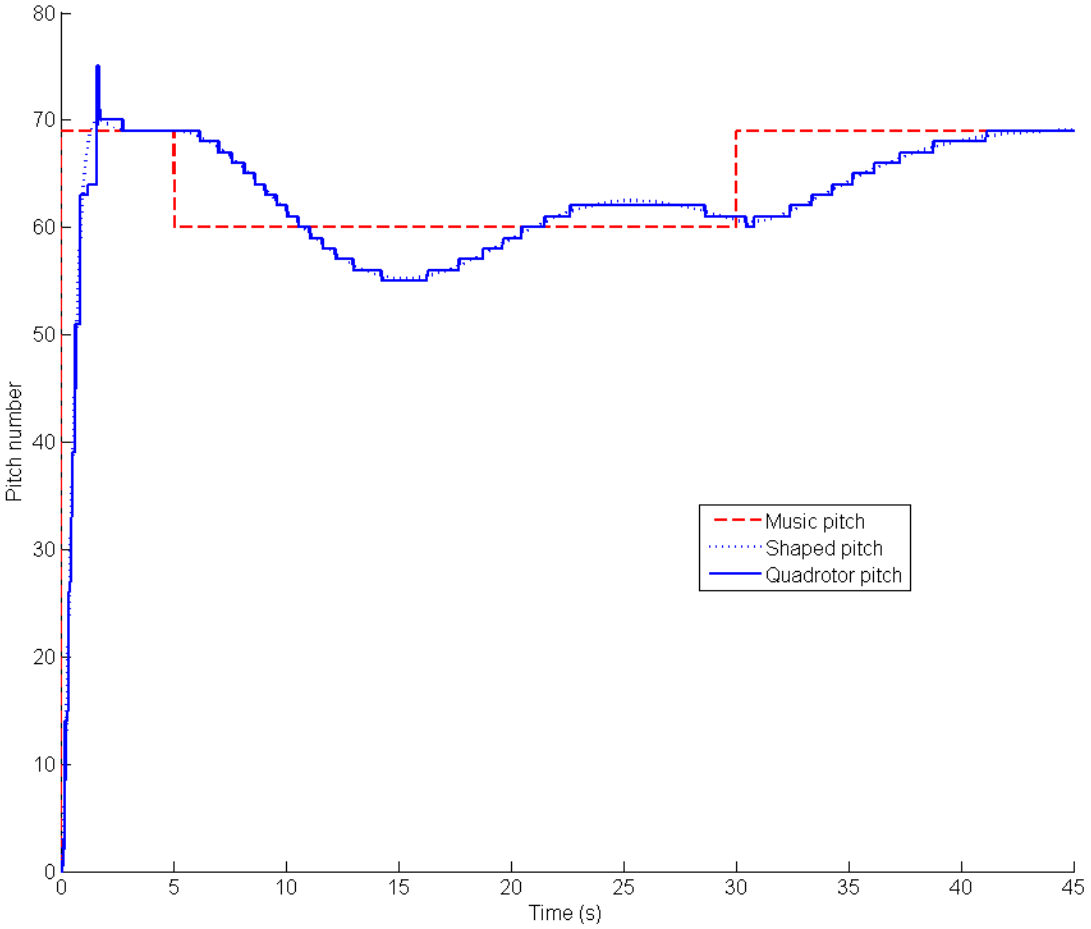


FIGURE 4.12: Changes of the pitch, the shaped pitch and the pitch played by the quadrotor.

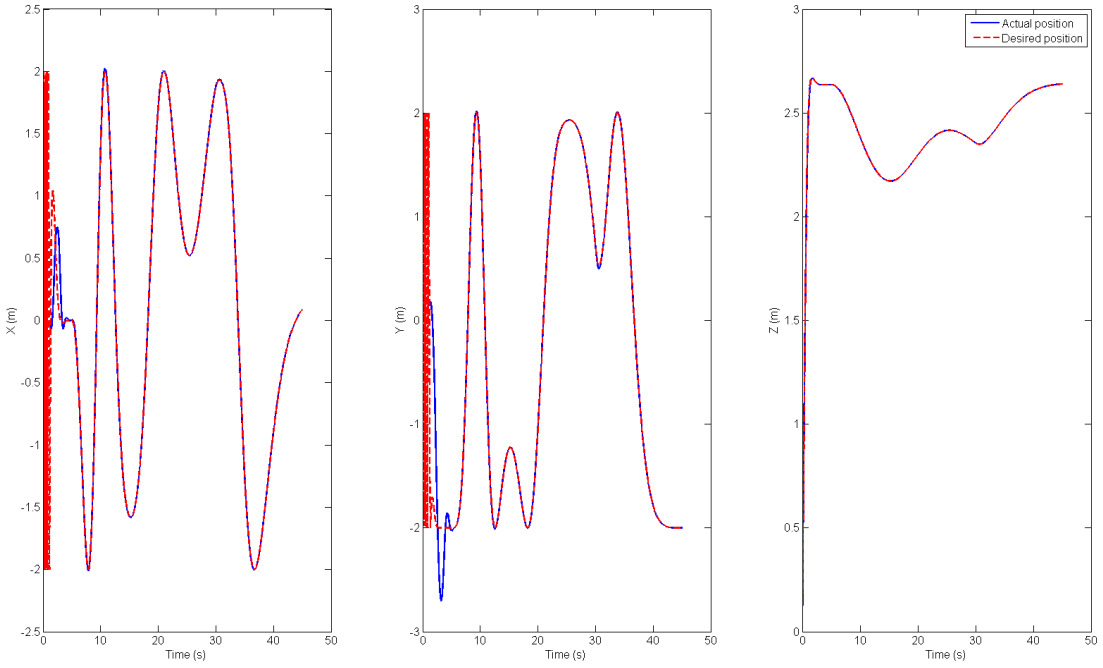


FIGURE 4.13: Position of the quadrotor and the desired trajectory.

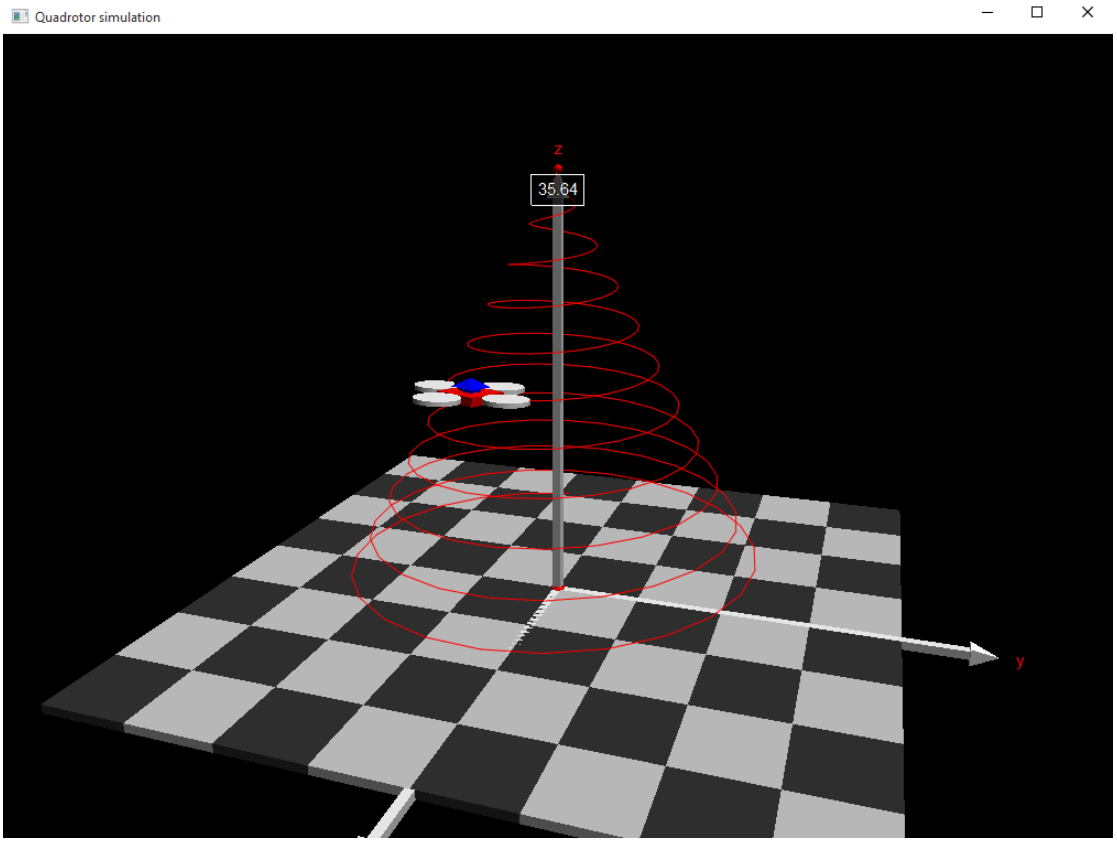


FIGURE 4.14: Screen capture of the quadrotor simulation with the cone motif

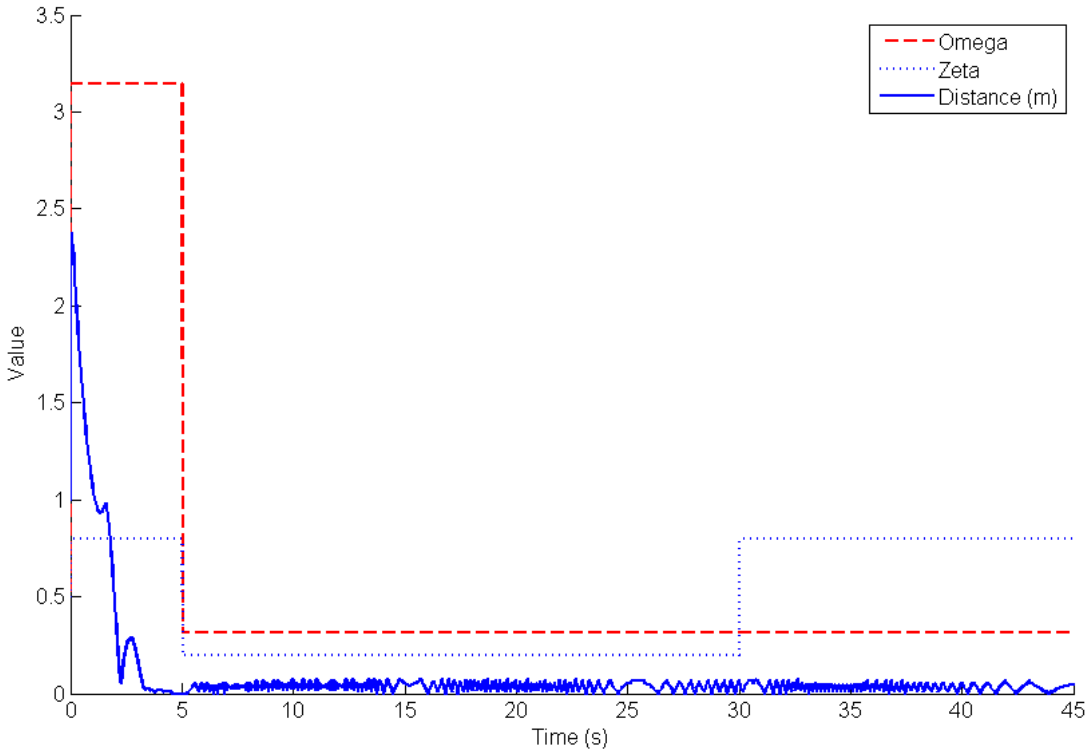


FIGURE 4.15: Changes of the shape parameters (the damping ratio and the natural frequency) and the distance as a function of time.

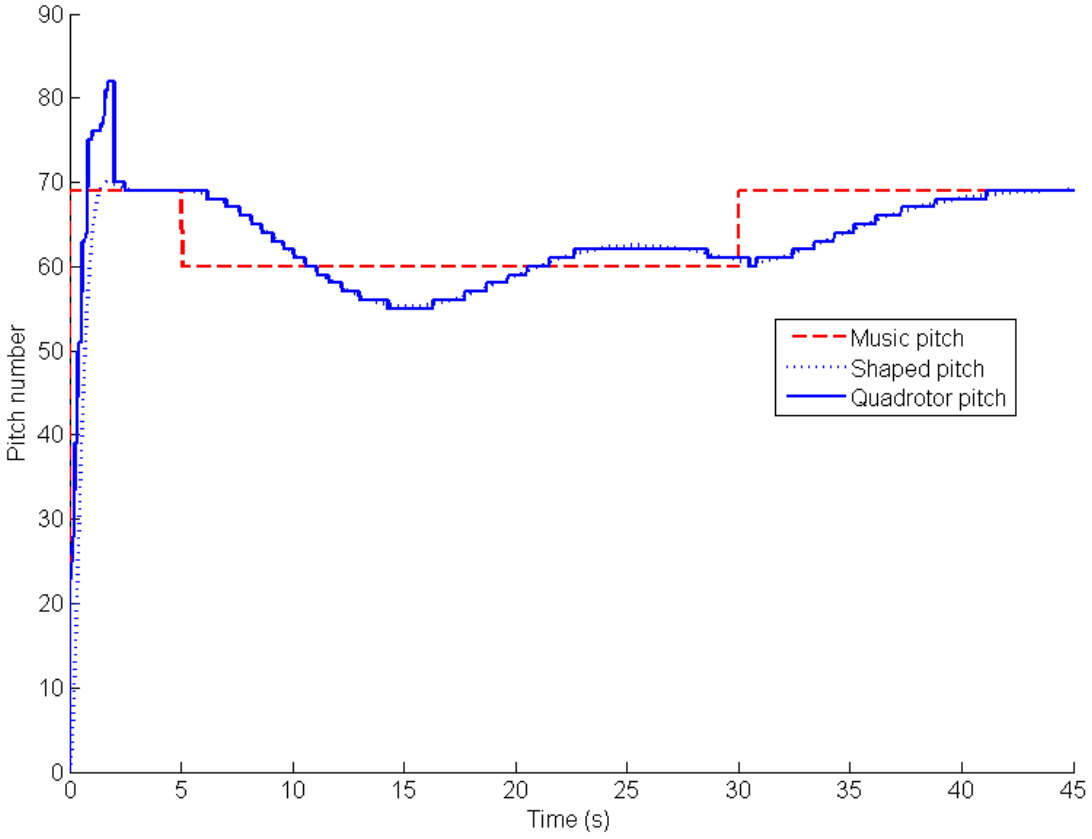


FIGURE 4.16: Changes of the pitch, the shaped pitch and the pitch played by the quadrotor.

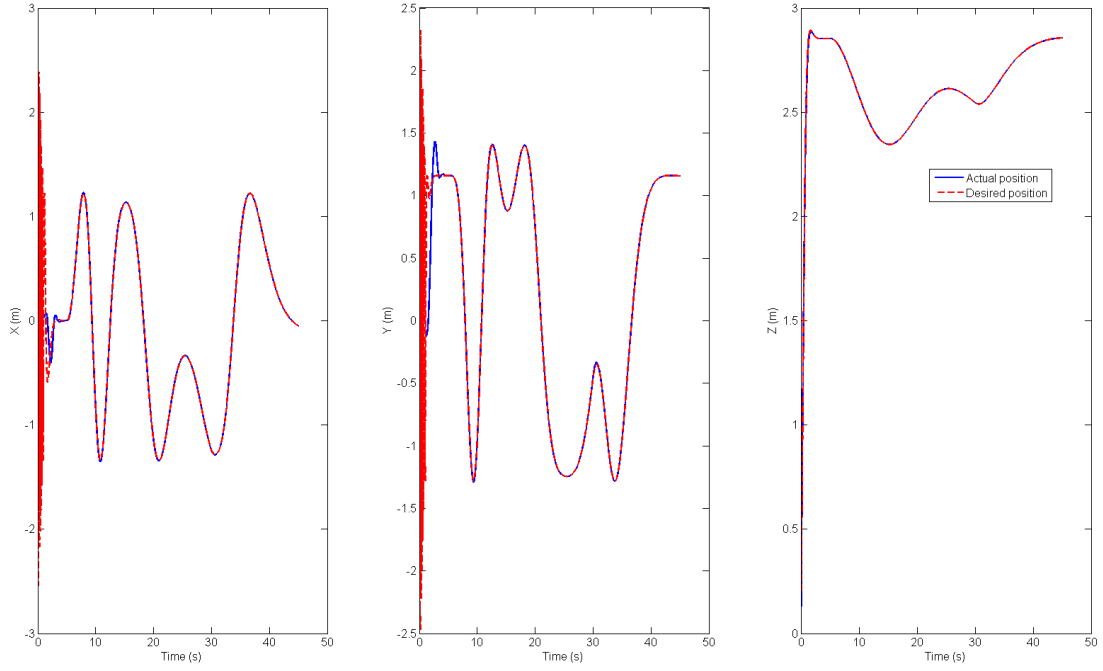


FIGURE 4.17: Position of the quadrotor and the desired trajectory.

Chapter 5

Extensions and conclusions

This chapter is the final chapter of the thesis, and includes two sections. The first section presents possible extensions to this work. The second section consists of a summary of the thesis, along with some concluding remarks.

5.1 Extensions

Several extensions to the system are proposed. First, the beat of the music is presented for potential use in the quadrotor choreography system. Second, an extension to multiple quadrotors is presented.

5.1.1 Beat of the music

In music and music theory, the beat is the basic unit of time [121]. It is the unit of time at which humans tap their feet when listening to music. The beat of the music can either be given a-priori by the musician (as done many times in a live performance) or it can be estimated by an off-the-shelf beat extraction software (see for example references [84, 88, 99]). The beat can be expressed in beats per minute, and in Hz.

The beat can be used in combination with the space component of the choreography. Let ω_b be the frequency of the beat, in rad/s. A simple path motif can be presented in which the concept of higher pitch is maintained, while introducing the notion of beat. While listening to the beat, the quadrotor would sway from side to side. In reference [10], quadrotors sway from side to side at an offline beat detection, where a phase-locked loop synchronizes the beat of the music and the periodic motion. As an extension, the pitch of the music being played can be associated to a particular height, similar to the

straight line mapping. This mapping can be described by equations (5.1), where h_{max} and h_{min} are the maximum and minimum allowable flight heights, N is the total number of allowable MIDI pitches, P is the pitch of the music currently being played, t is the time, and L is the sway length:

$$\begin{aligned} x &= L\cos(\omega_b t) \\ y &= 0 \\ z &= (h_{max} - h_{min})\frac{P}{N} + h_{min} \end{aligned} \tag{5.1}$$

5.1.2 Multiple quadrotors and multiple artists

In order to extend the mathematical formulation of a choreography to multiple quadrotors, several methods can be applied, depending on the desire of the artist(s). The system should include an obstacle avoidance algorithm.

Should the artist require multiple quadrotors to behave in the same way, different flying spaces can be assigned to each quadrotor. All the quadrotor displacements will be the same, and the performance will be identical in movement. Mathematically speaking, this requires an offset in any direction from the origin of the flying space of each quadrotor, and, of course, of the desired trajectories.

In the presence of multiple artists performing at the same time, different quadrotors can "listen" to different artists. In a simple implementation, different quadrotors will be receiving different pitch inputs. The quadrotors would be in different interchangeable flying spaces. Several parameters could be modified from one quadrotor to another, such as the path motif, or the shape filter parameters. An example would be a music band in a concert; the band could have a different quadrotor assigned to each musician. The quadrotor is assigned the flying space over the performer, such that the flying space offset is as dynamic as the movement of the performer on stage.

5.2 Concluding remarks

This thesis presented a novel systematic methodology to perform guidance navigation and control of a quadrotor in response to real-time music commands from a musician, based on the main components of a choreography. The methodology uses only one parameter extracted from musical features, and two parameters can be modified in real-time by an artist, to send guidance commands to the quadrotor allowing an effective real-time interaction with the musician.

Reference [122] is a recording of the first quadrotor performance designed by an artist, entitled *Geometry of Curves: Prelude on a Ramped Spiral.* This performance uses the mathematical formulation of a choreography and the elements presented in this thesis. In this performance, The music played by the artist is muted, and the quadrotor music playback sounds are generated in Ableton Live, a software for music production [123]. The MIDI information is sent to Ableton Live through a virtual port. The virtual port is managed by ipMIDI [124].

The four main components of a choreography are the space a dancer occupies, the shape of the body of the dancer, the structure of a dance with relation to the main timeline, and the timing of the dance. These four components were mathematically formulated for a quadrotor choreography. For the space component, different path motifs, parameterized by music, were presented: a straight line, a circle, and three helicoidal variations. For the shape component, a shapeless quadrotor is given movement characteristics by filtering the pitch through a second order filter to generate a shaped pitch. The parameters of this filter can be modified in real-time by the musician. The structure of the choreography is modified by a switching algorithm based on chord sequences, or by a pedal board and button-pedals. The time component refers to the dynamic equations describing the motion of the quadrotor, and the associated control system. The results show that the methodology for real-time guidance navigation and control in response to improvised music works effectively although it may present overshoot and some oscillation.

The quadrotor system allows for real-time control with an artist-in-the-loop. It is a system with minimal delays, in which the musician is given access to parameters that modify the performance:

- The pitch of the music the musician is playing.
- The damping ratio of the shape filter.
- The natural frequency of the shape filter.

The pitch of the music is designed by the artist, and can have a specific mood associated to it. A survey shows that the two parameters that control the motion of the quadrotor are intuitive, such that specific responses were expected by the survey participants for different musical moods. These responses have been associated to the changes caused by changes in the two parameters, the damping ratio and the natural frequency. Prior to the performance, several parameters can be modified such as the path motifs or the chord sequences.

Future research could focus on smoothing out the overshoot and oscillations by including a state predictor to compensate for delays. Future research could also include finding artistic and innovating mappings between music and quadrotor trajectories.

Appendix A

chordDetection.cpp

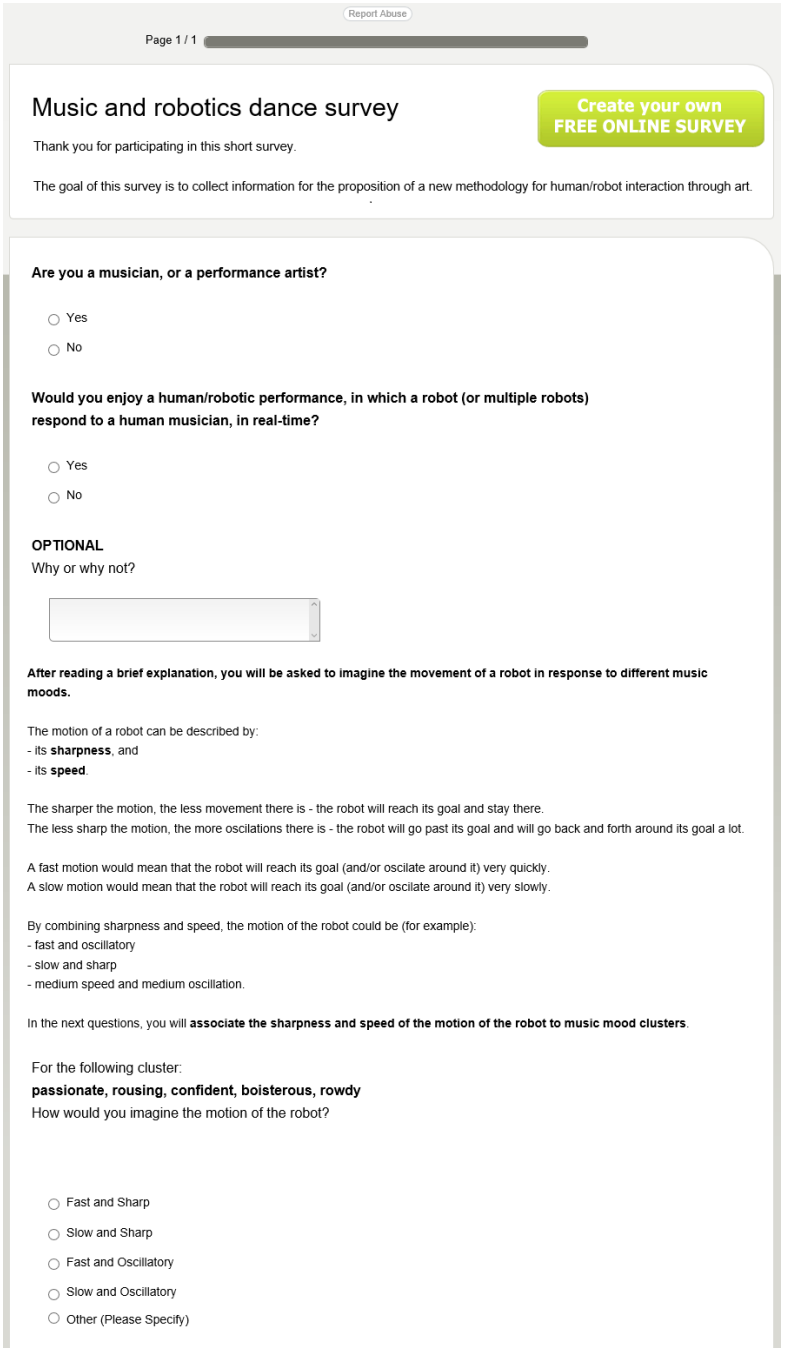
```
#include <iostream>

using namespace std;

int main()
{
        //changes depending on the pitches received;
        int number_of_pitches = 3;
        int pitches_received[3] = { 7,4,13}; // any positive integers
        int pitches[12] = { 0, 0, 0 ,0,0,0,0,0,0,0,0,0};
        int differences[3] = { 0, 0, 0 };

        //chords in semi-tone differences, recircling
        //major example: in semitones, it's 1-5-8
        //in differences, it's 4-3, and since it wraps around,
        //to detect, for example, an A major, pitch starts at 10
        //instead of 1

        //major
        int chords[4][3] = { { 4, 3, 5 }, //major
        { 3, 4, 5 },//minor
        { 4, 4, 4 },//aug
        { 3, 3, 6 } };//dim

        //wrap it around
        for (int i = 0; i < number_of_pitches; i++)
```

```
{
        pitches[pitches_received[i] % 12] = 1;
}

//get differences array
int temp1=-1, temp2=-1;
for (int i = 0; i < 12; i++)
{
        if (pitches[i] == 1)
        {
                if (temp1 != -1)
                {
                        temp2 = i;
                        for (int j = 0; j < 2; j++)
                        {
                                if (differences[j] == 0)
                                {
                                        differences[j] = temp2 - temp1;
                                        break;
                                }
                        }
                        temp1 = i;
                }
                else
                {
                        temp1 = i;
                }
        }
}
differences[2] = 12 - differences[0] - differences[1];

for (int chord_test = 0; chord_test < 4; chord_test++)
{
        for (int offset = 0; offset < 3; offset++)
        {
                bool test[3] = { false, false, false };
                for (int i = 0; i < 3; i++)
                {
                        if (chords[chord_test][i] == differences[((i + offset
```

```
                                {
                                        test[i] = true;
                                }
                                else
                                {
                                        for (int i = 0; i < 3; i++)
                                        {
                                                test[i] = false;
                                        }
                                        break;
                                }
                                if ((test[0] == true) && (test[1] = true) && (test[2]
                                {
                                        cout << "Chord found: ";
                                        switch (chord_test){
                                        case 0:
                                                cout << "major" << endl;
                                                break;
                                        case 1:
                                                cout << "minor" << endl;
                                                break;
                                        case 2:
                                                cout << "augmented" << endl;
                                                break;
                                        case 3:
                                                cout << "diminishied" << endl;
                                                break;
                                        }
                                        return 0;
                                }
                        }
                }
        }
        cout << "no chord found" << endl;
        return 0;
}
```

Appendix B

Survey questions



Report Abuse

Page 1 / 1

Music and robotics dance survey

Thank you for participating in this short survey.

The goal of this survey is to collect information for the proposition of a new methodology for human/robot interaction through art.

Are you a musician, or a performance artist?

- ○ Yes
- ○ No

Would you enjoy a human/robotic performance, in which a robot (or multiple robots) respond to a human musician, in real-time?

- ○ Yes
- ○ No

OPTIONAL
Why or why not?

After reading a brief explanation, you will be asked to imagine the movement of a robot in response to different music moods.

The motion of a robot can be described by:
- its **sharpness**, and
- its **speed**.

The sharper the motion, the less movement there is - the robot will reach its goal and stay there.
The less sharp the motion, the more oscilations there is - the robot will go past its goal and will go back and forth around its goal a lot.

A fast motion would mean that the robot will reach its goal (and/or oscilate around it) very quickly.
A slow motion would mean that the robot will reach its goal (and/or oscilate around it) very slowly.

By combining sharpness and speed, the motion of the robot could be (for example):
- fast and oscillatory
- slow and sharp
- medium speed and medium oscillation.

In the next questions, you will **associate the sharpness and speed of the motion of the robot to music mood clusters**.

For the following cluster:
passionate, rousing, confident, boisterous, rowdy
How would you imagine the motion of the robot?

- ○ Fast and Sharp
- ○ Slow and Sharp
- ○ Fast and Oscillatory
- ○ Slow and Oscillatory
- ○ Other (Please Specify)

For the following cluster:
rollicking, cheerful, fun, sweet, amiable/good natured
How would you imagine the motion of the robot?

- ○ Fast and Sharp
- ○ Slow and Sharp
- ○ Fast and Oscillatory
- ○ Slow and Oscillatory
- ○ Other (Please Specify)

For the following cluster:
literate, poignant, wistful, bittersweet, autumnal, brooding
How would you imagine the motion of the robot?

- ○ Fast and Sharp
- ○ Slow and Sharp
- ○ Fast and Oscillatory
- ○ Slow and Oscillatory
- ○ Other (Please Specify)

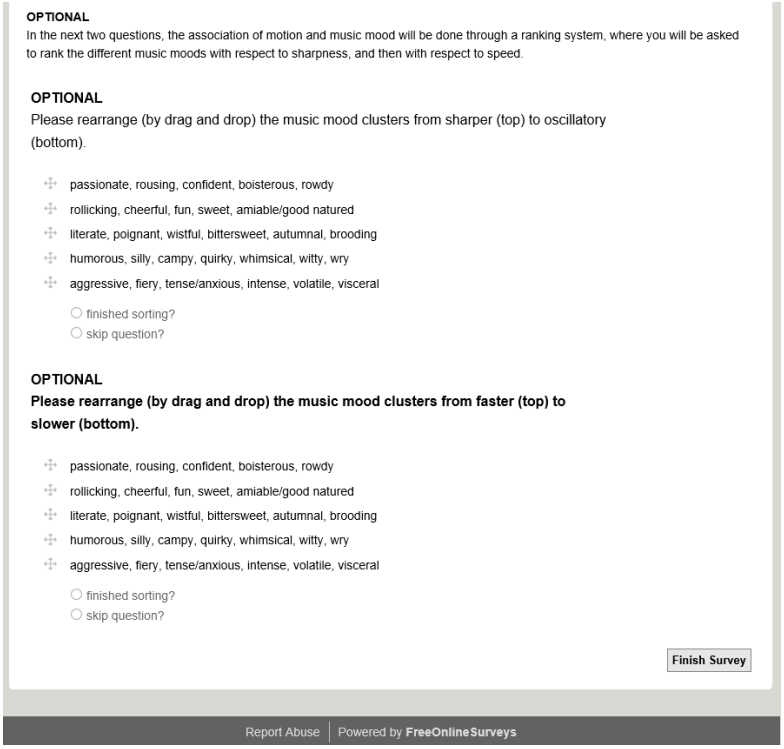
For the following cluster:
humorous, silly, campy, quirky, whimsical, witty, wry
How would you imagine the motion of the robot?

- ○ Fast and Sharp
- ○ Slow and Sharp
- ○ Fast and Oscillatory
- ○ Slow and Oscillatory
- ○ Other (Please Specify)

For the following cluster:
aggressive, fiery, tense/anxious, intense, volatile, visceral
How would you imagine the motion of the robot?

- ○ Fast and Sharp
- ○ Slow and Sharp
- ○ Fast and Oscillatory
- ○ Slow and Oscillatory
- ○ Other (Please Specify)

OPTIONAL
Would you be willing to control a robot through music at Concordia University? We would love it if you could come try out our system. If you are interested, please leave your email.



OPTIONAL
In the next two questions, the association of motion and music mood will be done through a ranking system, where you will be asked to rank the different music moods with respect to sharpness, and then with respect to speed.

OPTIONAL
Please rearrange (by drag and drop) the music mood clusters from sharper (top) to oscillatory (bottom).

- passionate, rousing, confident, boisterous, rowdy
- rollicking, cheerful, fun, sweet, amiable/good natured
- literate, poignant, wistful, bittersweet, autumnal, brooding
- humorous, silly, campy, quirky, whimsical, witty, wry
- aggressive, fiery, tense/anxious, intense, volatile, visceral

finished sorting?
skip question?

OPTIONAL
Please rearrange (by drag and drop) the music mood clusters from faster (top) to slower (bottom).

- passionate, rousing, confident, boisterous, rowdy
- rollicking, cheerful, fun, sweet, amiable/good natured
- literate, poignant, wistful, bittersweet, autumnal, brooding
- humorous, silly, campy, quirky, whimsical, witty, wry
- aggressive, fiery, tense/anxious, intense, volatile, visceral

finished sorting?
skip question?

Finish Survey

Report Abuse | Powered by **FreeOnlineSurveys**

Appendix C

Survey results

Are you a musician, or a performance artist?	Would you enjoy a human/robotic performance?	For the following cluster: passionate, rousing, confident, boisterous, rowdy How would you imagine the motion of the robot?	For the following cluster: rollicking, cheerful, fun, sweet, amiable/good natured How would you imagine the motion of the rob	For the following cluster: literate, poignant, wistful, bittersweet, autumnal, brooding How would you imagine the motion of the robot?	For the following cluster: humorous, silly, campy, quirky, whimsical, witty, wry How would you imagine the motion of the rob	For the following cluster: aggressive, fiery, tense/anxious, intense, volatile, visceral How would you imagine the motion of the robot?
Yes	No	Fast and Oscillatory	Slow and Sharp	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Fast and Oscillatory	Slow and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
No	Yes	Fast and Sharp	Slow and Sharp	Fast and Sharp	Slow and Oscillatory	Fast and Oscillatory
No	Yes	Fast and Oscillatory	Slow and Oscillatory	Slow and Oscillatory	Fast and Sharp	Fast and Sharp
No	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Fast and Oscillatory	Fast and Sharp	Slow and Oscillatory	Slow and Sharp	Fast and Oscillatory
No	Yes	Fast and Sharp	Fast and Oscillatory	Slow and Sharp	Slow and Oscillatory	Fast and Sharp
No	No	Fast and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Fast and Sharp	Slow and Sharp	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Fast and Sharp	Fast and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Oscillatory
No	Yes	Fast and Oscillatory	Slow and Sharp	Slow and Oscillatory	Medium medium	Fast and Sharp
Yes	Yes	Slow and Oscillatory	Fast and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Sharp	Fast and Oscillatory
No	Yes	Fast and Sharp	Fast and Sharp	Fast and Oscillatory	Slow and Sharp	Fast and Sharp
No	No	Slow and Oscillatory	Fast and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Sharp
No	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Oscillatory
No	No	Fast and Sharp	Fast and Oscillatory	Slow and Oscillatory	Slow and Sharp	Slow and Oscillatory
No	Yes	Slow and Sharp	Fast and Oscillatory	Fast and Sharp	Slow and Oscillatory	Slow and Sharp
No	Yes	Fast and Oscillatory	Slow and Sharp	Slow and Oscillatory	Fast and Sharp	Slow and Oscillatory
No	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Sharp	Fast and Sharp	Fast and Sharp
Yes	No	Fast and Oscillatory	Fast and Sharp	Slow and Oscillatory	Slow and Sharp	Fast and Sharp
No	No	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Sharp	Slow and Oscillatory
Yes	Yes	Fast and Oscillatory	Slow and Oscillatory	Slow and Sharp	Fast and Sharp	Slow and Sharp
No	Yes	Slow and Sharp	Fast and Sharp	Slow and Oscillatory	Fast and Sharp	Fast and Sharp
No	No	Fast and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
No	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
No	Yes	Fast and Oscillatory	Slow and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Sharp
No	Yes	Fast and Oscillatory	Slow and Oscillatory	Fast and Sharp	Slow and Oscillatory	Fast and Sharp
No	Yes	Fast and Sharp	Fast and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Sharp
No	Yes	Fast and Oscillatory	Slow and Oscillatory	Slow and Sharp	Medium speed and medi	Fast and Sharp
Yes	Yes	Slow and Sharp	Slow and Sharp	Fast and Sharp	Fast and Sharp	Fast and Oscillatory
No	Yes	Slow and Sharp	Fast and Sharp	Slow and Oscillatory	Fast and Oscillatory	Fast and Oscillatory
No	Yes	Fast and Sharp	Fast and Sharp	Fast and Sharp	Fast and Sharp	Fast and Sharp
Yes	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Sharp
No	Yes	Fast and Oscillatory	Fast and Sharp	Fast and Oscillatory	Fast and Oscillatory	Fast and Sharp
No	No	Slow and Sharp	Fast and Oscillatory	Fast and Oscillatory	Slow and Sharp	Slow and Sharp
No	Yes	Fast and Oscillatory		Fast and Sharp	Slow and Oscillatory	Fast and Sharp
Yes	Yes	Fast and Sharp	Fast and Sharp	Fast and Sharp	Fast and Sharp	Fast and Sharp
Yes	Yes	Fast and Oscillatory	Slow and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Sharp
No	No	Slow and Sharp	Slow and Sharp	Slow and Sharp	Slow and Sharp	Slow and Sharp
Yes	Yes	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp	Fast and Oscillatory	Fast and Sharp
No	No	Slow and Sharp	Slow and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
No	Yes	Fast and Sharp	Fast and Oscillatory	Slow and Sharp	Slow and Oscillatory	Slow and Oscillatory
No	No	Slow and Sharp	Slow and Sharp	Slow and Sharp	Slow and Sharp	Slow and Sharp
No	Yes	Fast and Oscillatory	Slow and Oscillatory	Fast and Sharp	Fast and Oscillatory	Fast and Sharp
No	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
Yes	No	Slow and Sharp	Fast and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Oscillatory
No	Yes	Fast and Sharp	Fast and Oscillatory	Slow and Sharp	Slow and Oscillatory	Fast and Sharp
Yes	Yes	Fast and Sharp	Fast and Oscillatory	Slow and Oscillatory	Slow and Oscillatory	Fast and Sharp
Yes	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	Slow and Oscillatory	Fast and Sharp
Yes	Yes	Fast and Oscillatory	Slow and Sharp	Slow and Oscillatory	Fast and Sharp	Fast and Sharp
Yes	Yes	Slow and Oscillatory	Fast and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Oscillatory
Yes	Yes	Fast and Oscillatory	Fast and Sharp	Slow and Sharp	Fast and Oscillatory	Fast and Sharp
Yes	Yes					
Yes	Yes	Fast and Oscillatory	Fast and Sharp	Slow and Oscillatory	Fast and Sharp	Slow and Sharp
Yes	Yes	Fast and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Sharp	Fast and Sharp
Yes	Yes	Slow and Oscillatory	Fast and Oscillatory	Slow and Sharp	Slow and Oscillatory	Fast and Oscillatory
Yes	Yes	Fast and Oscillatory	Fast and Sharp	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Slow and Oscillatory	Fast and Oscillatory	Slow and Oscillatory	might need to be asymm	Fast and Sharp
No	Yes	Fast and Oscillatory	Slow and Oscillatory	Slow and Sharp	Slow and Oscillatory	Fast and Sharp
Yes	Yes	Fast and Oscillatory	Slow and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Fast and Sharp	Fast and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Oscillatory
Yes	No	Fast and Oscillatory	Slow and Oscillatory	Slow and Sharp	Slow and Oscillatory	Fast and Sharp
No	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	slow/sharp and random	Fast and Sharp
Yes	Yes	Fast and Sharp	Slow and Oscillatory	Slow and Sharp	Fast and Oscillatory	Fast and Oscillatory
Yes	Yes					
Yes	No	Slow and Sharp	Fast and Sharp	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp
Yes	Yes	Slow and Sharp	Fast and Oscillatory	Slow and Oscillatory	Fast and Oscillatory	Fast and Sharp

Appendix D

main.py

```
from visual import *
from model import *
from visual.graph import *          # import graphing features

def within(x, lowerRange, higherRange):
    if ( x > lowerRange and x < higherRange):
        return True
    else:
        return False

################################################ Graphics
## declare scene and camera orientation
scene = display(title='Quadrotor simulation',
     x=0, y=0, z=0, width=1024, height=768,
     center=(0,0,2), up = (0,0,1), forward = (-3,-1,-1))


## axes and floor
checkerboard = (  (0.2,0.8,0.2,0.8,0.2,0.8,0.2,0.8),
                  (0.8,0.2,0.8,0.2,0.8,0.2,0.8,0.2),
                  (0.2,0.8,0.2,0.8,0.2,0.8,0.2,0.8),
                  (0.8,0.2,0.8,0.2,0.8,0.2,0.8,0.2),
                  (0.2,0.8,0.2,0.8,0.2,0.8,0.2,0.8),
                  (0.8,0.2,0.8,0.2,0.8,0.2,0.8,0.2),
                  (0.2,0.8,0.2,0.8,0.2,0.8,0.2,0.8),
                  (0.8,0.2,0.8,0.2,0.8,0.2,0.8,0.2)
```

```
                    )
tex = materials.texture(data=checkerboard,
                        mapping="rectangular",
                        interpolate=False)
floor = box( size = (0.1,8,8), material = tex, axis = (0,0,1))
xaxis = arrow(pos = (0,0,0), axis = (5,0,0), title = "x", shaftwidth=0.1)
yaxis = arrow(pos = (0,0,0), axis = (0,5,0), title = "y", shaftwidth=0.1)
zaxis = arrow(pos = (0,0,0), axis = (0,0,5), title = "z", shaftwidth=0.1)
label(text = "x", pos = (5.2,0,0), opacity = 0, box=0, line=0, color = color.red)
label(text = "y", pos = (0,5.2,0), opacity = 0, box=0, line=0, color = color.red)
label(text = "z", pos = (0,0,5.2), opacity = 0, box=0, line=0, color = color.red)

## quadrotor graphics
quadrotor = frame(make_trail=False)
box (frame = quadrotor, size = (0.5,0.5,0.1), axis = (0,0,-1), color = color.red)
cylinder (frame = quadrotor, radius = 0.25, axis = (0.05,0,0), pos = (0,-0.5,0))
cylinder (frame = quadrotor, radius = 0.25, axis = (0.05,0,0), pos = (0,0.5,0))
cylinder (frame = quadrotor, radius = 0.25, axis = (0.05,0,0), pos = (0,0,0.5))
cylinder (frame = quadrotor, radius = 0.25, axis = (0.05,0,0), pos = (0,0,-0.5))
arrow(frame = quadrotor, axis = (0,0,-0.25), pos = (0.1,0,0), shaftwidth = 0.05, colo
arrow(frame = quadrotor, axis = (0.2,0,0), shaftwidth = 0.15, color = color.blue)
quadrotor.axis = (0,0,1)
## desired position and waypoint position
waypointPosition = sphere(pos=(0,0,0), radius=0.1, color=color.blue, make_trail = Fal

## Simulation time and display
simulationTime = 0
lbl = label(yoffset=350, line=0)

################################################ quadrotor initial conditions
x = 0
y = 0
z = 0
xdot = 0
ydot = 0
zdot = 0
xddot = 0
yddot = 0
zddot = 0
```

```
yaw = 0
roll = 0
pitch = 0
yawdot = 0
rolldot = 0
pitchdot = 0

## desired
x_waypoint = 0
y_waypoint = 0
z_waypoint = 2

################################################ TIME (Controller input)
from controllers import *
# declare position controller
control = Control()

################################################ SHAPE (second order filter)
from shape import *
###declare motion controller
shapeFilter = ShapeControl()

################################################ Listen to MIDI process through TCP
from TCP import *
## declare TCP thread
##listener = ListenToTCP(socket.getfqdn(),5005)
listener = ListenToTCP('127.0.0.1',5005)
##listener = ListenToTCP('192.168.1.2',5005)
listener.daemon = True
listener.start();



################################################ SPACE and STRUCTURE (path motifs and
motifSphere = sphere(radius = 0.05, color = color.red, make_trail = True, retain=127*
from motifs import motif
from motifs import inverseMotif

## music parameters
```

```
musicPitch = 0
shapedPitch = 0
drawMotifPitch = 0
dancing = False
motifNumber = 1
updateMotif = True
redrawMotif = True

def changeMotifNumberTo(number):
    global motifNumber
    global redrawMotif
    if not(number == motifNumber):
        motifNumber = number;
        global updateMotif
        updateMotif = True
        redrawMotif = True


################################################## MIDI playback
playMusic = True
if(playMusic):
    import pygame.midi
    pygame.mixer.pre_init(22050, -16, 1, 512)
    pygame.init()
    pygame.fastevent.init()
    event_get = pygame.fastevent.get
    event_post = pygame.fastevent.post

    pygame.midi.init()

    ##  1-8        Piano                                65-72         Reed
    ##  9-16        Chromatic Percussion        73-80        Pipe
    ##  17-24        Organ                          81-88        Synth Lead
    ##  25-32        Guitar                          89-96        Synth Pad
    ##  33-40        Bass                          97-104        Synth Effects
    ##  41-48        Strings                          105-112        Ethnic
    ##  49-56        Ensemble                113-120        Percussive
    ##  57-64 Brass                        121-128 Sound Effects
    instrument = 17
```

```
    midi_out = pygame.midi.Output(pygame.midi.get_default_output_id(), 0)
    midi_out.set_instrument(instrument)
    quadrotorMusicPitch = 60
    quadrotorMusicAmplitude = 127


    playRate = 0.05
    shapeFilter.omega = 0.5
    lastPlayTime = 0


################################################ LOOP
musicPitch = 100
while True:

    ## draw motif (trail of motifSphere)
    if updateMotif == True:
        if(redrawMotif):
            drawMotifPitch = 0
            redrawMotif = False
        if (drawMotifPitch > 127):
            drawMotifPitch = 0
            updateMotif = False
        else:
            motifSphere.pos = motif(motifNumber,drawMotifPitch)
            drawMotifPitch += .5

    if (simulationTime < 5):
        changeMotifNumberTo(2)
        musicPitch = 69
        shapeFilter.zeta = 0.707
        shapeFilter.omega = 1*pi
    elif (simulationTime < 10):
        musicPitch = 60
        shapeFilter.zeta = 0.307
    elif (simulationTime < 15):
        changeMotifNumberTo(1)
    elif (simulationTime < 30):
        shapeFilter.omega = 0.1*pi
```

```
        musicPitch = 69
    elif (simulationTime < 45):
        musicPitch = 60
        shapeFilter.zeta = 0.8
        shapeFilter.omega = 0.2*pi
    elif (simulationTime < 60):
        writeFile = False

    ## trajectory motifs
    if(dancing == True):
        x_waypoint,y_waypoint,z_waypoint = motif(motifNumber, shapedPitch)

    shapedPitch = shapeFilter.update(musicPitch)

    control.update(x,y,z,x_waypoint,y_waypoint,z_waypoint, roll, pitch, yaw)
    simulationTime += model.dt
    rate(1/(model.dt))



    F1 = control.U1/4 - control.U2/(4*model.l) + control.U3/(4*model.l)# - control.U4
    F2 = control.U1/4 - control.U2/(4*model.l) - control.U3/(4*model.l)# + control.U4
    F3 = control.U1/4 + control.U2/(4*model.l) + control.U3/(4*model.l)# + control.U4
    F4 = control.U1/4 + control.U2/(4*model.l) - control.U3/(4*model.l)# - control.U4

    ## linear dynamics
    xddot = (sin(yaw)*roll + cos(yaw)*pitch)*(F1 + F2 + F3 + F4)
    yddot = (sin(yaw)*pitch - cos(yaw)*roll)* (F1 + F2 + F3 + F4)
    zddot = (F1 + F2 + F3 + F4)/mass - g
    rollddot = model.l*(-F1 - F2 + F3 + F4)/Ixx
    pitchddot = model.l*(+F1 - F2 + F3 - F4)/Iyy
    yawddot = model.kYaw * (-F1 + F2 + F3 - F4)/Izz

    ## integration
    xdot = xdot + xddot*model.dt
    x = x + xdot*model.dt
    ydot = ydot + yddot*model.dt
    y = y + ydot*model.dt
    zdot = zdot + zddot*model.dt
    z = z + zdot*model.dt
```

```
rolldot = rolldot + rollddot*model.dt
roll = roll + rolldot*model.dt
pitchdot = pitchdot + pitchddot*model.dt
pitch = pitch + pitchdot*model.dt
yawdot = yawdot + yawddot*model.dt
yaw = yaw + yawdot*model.dt

## update graphics
quadrotor.pos = (x,y,z)
quadrotor.rotate(angle = yawdot*model.dt)
quadrotor.axis = (sin(yaw)*sin(roll) + cos(yaw)*sin(pitch)*cos(roll),sin(yaw)*sin
waypointPosition.pos = (x_waypoint, y_waypoint, z_waypoint)


## check TCP thread for updates
if not listener.q.empty():
    listenerData = listener.q.get()
    if listenerData[0] == "x":
        x_waypoint = float(listenerData[1:]) - 50
        dancing = False
    if listenerData[0] == "y":
        y_waypoint = float(listenerData[1:]) - 50
        dancing = False
    if listenerData[0] == "z":
        z_waypoint = float(listenerData[1:]) - 50
        dancing = False
    if listenerData[0] == "d":
        shapeFilter.zeta = (float(listenerData[1:]) - 50.0)
    if listenerData[0] == "w":
        shapeFilter.omega = (float(listenerData[1:]) - 50.0)
    if listenerData[0] == "p":
        dancing = True
        musicPitch = float(listenerData[1:])- 50
    if listenerData[0] == "m":
        dancing = True
        changeMotifNumberTo(int(listenerData[1:])- 50)


## Extract music pitch from quadrotor position
```

```
    dancing = True
    if(playMusic):
        if ((simulationTime < playRate) or (simulationTime - lastPlayTime > playRate)
            newPitch , quadrotorMusicAmplitude = inverseMotif(motifNumber, x, y, z)
##            pygame.mixer.Sound.set_volume(quadrotorMusicAmplitude/127)
##            if (not(quadrotorMusicPitch == newPitch)):
            if True:
                midi_out.note_off(int(quadrotorMusicPitch))
                quadrotorMusicPitch = newPitch
                midi_out.note_on(int(quadrotorMusicPitch),int(round(quadrotorMusicAmp
            lastPlayTime = simulationTime
```

Appendix E

controllers.py

```
from math import *
import model

def bound(value, low, high):
    return max(low, min(high, value))

writeFile = True

class Control:
    def __init__(self):

        ## controller gains
        ## attitude
        self.omegaAttitude = 2*3.141592
        self.zetaAttitude = 0.6
        ## position
        self.zetaZ = 0.6
        self.omegaZ = 3*3.141592
        self.zetaSides = 1.2
        self.omegaSides = 2*3.141592
        ## z variables
        self.z_camera = [0,0,0]
        self.z_waypoint = [0,0,0]
        self.z_ddot_ref = [0,0,0]
        ## horizontal variables
        self.x_camera = [0,0,0]
```

```
        self.x_waypoint = [0,0,0]
        self.x_ddot_ref = [0,0,0]
        self.y_camera = [0,0,0]
        self.y_waypoint = [0,0,0]
        self.y_ddot_ref = [0,0,0]
        # orientation variables
        self.roll_camera = [0,0,0]
        self.roll_waypoint = [0,0,0]
        self.roll_ddot_ref = [0,0,0]
        self.roll_error = [0,0,0]
        self.pitch_camera = [0,0,0]
        self.pitch_waypoint = [0,0,0]
        self.pitch_ddot_ref = [0,0,0]
        self.yaw_camera = [0,0,0]
        self.yaw_waypoint = [0,0,0]
        self.yaw_ddot_ref = [0,0,0]
        # control inputs
        self.U1 = 0
        self.U2 = 0
        self.U3 = 0
        self.U4 = 0
        self.U1d = 0
        self.U2d = 0
        self.U3d = 0
        self.U4d = 0
        self.U1dp = 0
        self.U2dp = 0
        self.U3dp = 0
        self.U4dp = 0
        self.tauU = 0.004

    def update(self,x = 0, y = 0, z = 0, xDesired = 0, yDesired = 0, zDesired = 0, ro

        self.U1dp = self.U1d
        self.U2dp = self.U2d
        self.U3dp = self.U3d
        self.U4dp = self.U4d
        # z calculations
        self.z_camera[2] = self.z_camera[1]
```

```
        self.z_camera[1] = self.z_camera[0]
        self.z_camera[0] = z
        self.z_waypoint[2] = self.z_waypoint[1]
        self.z_waypoint[1] = self.z_waypoint[0]
        self.z_waypoint[0] = zDesired
        self.z_ddot_ref[2] = self.z_ddot_ref[1]
        self.z_ddot_ref[1] = self.z_ddot_ref[0]
        k1 = self.omegaZ*self.omegaZ
        k2 = 2*self.zetaZ*self.omegaZ
        self.z_ddot_ref[0] = self.z_camera[0]*(k1 + 2*k2/model.dt) + self.z_camera[1]
        self.z_ddot_ref[0] *= -1
        # x calculations
        self.x_camera[2] = self.x_camera[1]
        self.x_camera[1] = self.x_camera[0]
        self.x_camera[0] = x
        self.x_waypoint[2] = self.x_waypoint[1]
        self.x_waypoint[1] = self.x_waypoint[0]
        self.x_waypoint[0] = xDesired
        self.x_ddot_ref[2] = self.x_ddot_ref[1]
        self.x_ddot_ref[1] = self.x_ddot_ref[0]
        k1 = self.omegaSides*self.omegaSides
        k2 = 2*self.zetaSides*self.omegaSides
        self.x_ddot_ref[0] = self.x_camera[0]*(k1 + 2*k2/model.dt) + self.x_camera[1]
        self.x_ddot_ref[0] *= -1
        # y calculations
        self.y_camera[2] = self.y_camera[1]
        self.y_camera[1] = self.y_camera[0]
        self.y_camera[0] = y
        self.y_waypoint[2] = self.y_waypoint[1]
        self.y_waypoint[1] = self.y_waypoint[0]
        self.y_waypoint[0] = yDesired
        self.y_ddot_ref[2] = self.y_ddot_ref[1]
        self.y_ddot_ref[1] = self.y_ddot_ref[0]
        k1 = self.omegaSides*self.omegaSides
        k2 = 2*self.zetaSides*self.omegaSides
        self.y_ddot_ref[0] = self.y_camera[0]*(k1 + 2*k2/model.dt) + self.y_camera[1]
        self.y_ddot_ref[0] *= -1

        # into angles and U1
```

```
self.U1d = model.mass*(self.z_ddot_ref[0] + model.g)
roll_desired = (model.mass/model.g)*(self.x_ddot_ref[0]*sin(yaw) - self.y_ddo
pitch_desired = (model.mass/model.g)*(self.x_ddot_ref[0]*cos(yaw) + self.y_dd
yaw_desired = yaw

## attitude controller
## bound desired angles
orientationLimit = pi/4
pitch_desired = bound(pitch_desired,-orientationLimit, orientationLimit)
roll_desired = bound(roll_desired, -orientationLimit, orientationLimit)

# roll calculations
self.roll_camera[2] = self.roll_camera[1]
self.roll_camera[1] = self.roll_camera[0]
self.roll_camera[0] = roll
self.roll_waypoint[2] = self.roll_waypoint[1]
self.roll_waypoint[1] = self.roll_waypoint[0]
self.roll_waypoint[0] = roll_desired
self.roll_ddot_ref[2] = self.roll_ddot_ref[1]
self.roll_ddot_ref[1] = self.roll_ddot_ref[0]
k1 = self.omegaAttitude*self.omegaAttitude
k2 = 2*self.zetaAttitude*self.omegaAttitude
self.roll_ddot_ref[0] = self.roll_camera[0]*(k1 + 2*k2/model.dt) + self.roll_
self.roll_ddot_ref[0] *= -1
self.roll_ddot_ref[0] = bound(self.roll_ddot_ref[0], -300, 300)

self.U2d = model.Ixx*self.roll_ddot_ref[0]


    # pitch calculations
self.pitch_camera[2] = self.pitch_camera[1]
self.pitch_camera[1] = self.pitch_camera[0]
self.pitch_camera[0] = pitch
self.pitch_waypoint[2] = self.pitch_waypoint[1]
self.pitch_waypoint[1] = self.pitch_waypoint[0]
self.pitch_waypoint[0] = pitch_desired
self.pitch_ddot_ref[2] = self.pitch_ddot_ref[1]
self.pitch_ddot_ref[1] = self.pitch_ddot_ref[0]
k1 = self.omegaAttitude*self.omegaAttitude
```

```
k2 = 2*self.zetaAttitude*self.omegaAttitude
self.pitch_ddot_ref[0] = self.pitch_camera[0]*(k1 + 2*k2/model.dt) + self.pit
self.pitch_ddot_ref[0] *= -1
self.pitch_ddot_ref[0] = bound(self.pitch_ddot_ref[0], -300, 300)
self.U3d = model.Iyy*self.pitch_ddot_ref[0]

# yaw calculations
self.yaw_camera[2] = self.yaw_camera[1]
self.yaw_camera[1] = self.yaw_camera[0]
self.yaw_camera[0] = yaw
self.yaw_waypoint[2] = yaw_desired
self.yaw_waypoint[1] = yaw_desired
self.yaw_waypoint[0] = yaw_desired
self.yaw_ddot_ref[2] = self.yaw_ddot_ref[1]
self.yaw_ddot_ref[1] = self.yaw_ddot_ref[0]
self.yaw_ddot_ref[0] = (4 * self.yaw_waypoint[2] - 8 * self.yaw_waypoint[1] +
self.yaw_ddot_ref[0] = bound(self.yaw_ddot_ref[0], -300, 300)
self.U4d = model.Izz*self.yaw_ddot_ref[0]

U1Limit = 42
self.U1d = bound(self.U1d, -U1Limit, U1Limit)
U23Limit = 80
self.U2d = bound(self.U2d, -U1Limit, U1Limit)
self.U3d = bound(self.U3d, -U1Limit, U1Limit)
self.U4d = bound(self.U4d, -U1Limit, U1Limit)

self.U1 = (model.dt*(self.U1dp + self.U1d) - (model.dt - 2 * self.tauU)*self.
self.U2 = (model.dt*(self.U2dp + self.U2d) - (model.dt - 2 * self.tauU)*self.
self.U3 = (model.dt*(self.U3dp + self.U2d) - (model.dt - 2 * self.tauU)*self.
self.U4 = (model.dt*(self.U4dp + self.U4d) - (model.dt - 2 * self.tauU)*self.
```

Appendix F

shape.py

```
import model

class ShapeControl:

    def __init__(self):

        # pitch variable
        self.musicPitch_output = [0,0,0]
        self.musicPitch_reference = [0,0,0]
        # music parameters
        self.zeta = 0.5
        self.omega = 0.9*3.141592


    def update(self,musicPitchDesired):
        # musicPitch calculations
        self.musicPitch_reference[2] = self.musicPitch_reference[1]
        self.musicPitch_reference[1] = self.musicPitch_reference[0]
        self.musicPitch_reference[0] = musicPitchDesired
        self.musicPitch_output[2] = self.musicPitch_output[1]
        self.musicPitch_output[1] = self.musicPitch_output[0]
        self.musicPitch_output[0] = (model.dt*model.dt*self.omega*self.omega*self.mus

        return self.musicPitch_output[0]
```

Appendix G

motifs.py

```
from math import *

def my_range(start, end, step):
    while start <= end:
        yield start
        start += step

def motif(motifNumber, pitch):
    Q = 12
    if motifNumber == 0:
        x = 0
        y = 0
        z = 1
    elif motifNumber == 1:
        ## helix
        x = 2 * cos(2*3.141592*pitch/12)
        y = 2 * sin(2*3.141592*pitch/12)
        z = 4.3*pitch/127.0 + 0.3

    elif motifNumber == 2:
        ## vertical line
        x = 0
        y = 0
        z = 4.7*pitch/127.0 + 0.3
    elif motifNumber == 3:
        ## horizontal line
```

```
        x = 0
        y = 4.7*pitch/127.0 + 0.3
        z = 2
    elif motifNumber == 4:
        ## circle of notes
        x = 2 * cos(2*3.141592*pitch/12)
        y = 2 * sin(2*3.141592*pitch/12)
        z = 2
    elif motifNumber == 5:
        ## cone
        x = (pitch-127)/50 * cos(2*3.141592*pitch/12)
        y = (pitch-127)/50 * sin(2*3.141592*pitch/12)
        z = (pitch-127)/30 + 5
        z = (pitch/127)*4.7 + 0.3
    elif motifNumber == 6:
        ## circles
        x = 2 * cos(2*3.141592*pitch/12)
        y = 2 * sin(2*3.141592*pitch/12)
        z = 4.7*(pitch - (pitch\%12) )/(12*11) + 0.3
    else:
        x = 0
        y = 0
        z = 0

    return x,y,z

def inverseMotif(motifNumber, x, y, z):
    distance = 100
    closestPitch = 1
    for numericalPitch in my_range(0,127,1):
        numX, numY, numZ = motif(motifNumber, numericalPitch)
        distanceTemp = sqrt ( (x - numX)**2 + (y - numY)**2 + (z - numZ)**2)
        if (distanceTemp < distance):
            distance = distanceTemp
            closestPitch = numericalPitch
    closestPitch = round(closestPitch)
    amplitude = round(-127*distance + 127)
    return closestPitch, amplitude
```

Appendix H

TCP.py

```
import socket
import threading
import Queue

class ListenToTCP(threading.Thread):
    def __init__(self, myIP, myPORT):
        super(ListenToTCP, self).__init__()
        self.myIP = myIP
        self.myPORT = myPORT
        self.q = Queue.Queue()
        self.sock = socket.socket(socket.AF_INET, # Internet
                    socket.SOCK_STREAM) # TCP
        self.sock.bind((myIP,myPORT)) #used for TCP
        self.data = ""

    def run(self):
        self.sock.listen(1)
        self.conn, self.addr = self.sock.accept()
        print "CONNECTION SUCCESSFUL TO ", self.addr
        while 1:
            self.data= self.conn.recv(6) # buffer size is 6 bytes
            self.q.put((self.data))
```

Appendix I

MIDI.py

```
import pygame
import pygame.midi
import socket
import Queue
import thread
MIDInotesON = [0] * 127

##def input_main(device_id = None):
pygame.init()
pygame.fastevent.init()
event_get = pygame.fastevent.get
event_post = pygame.fastevent.post

pygame.midi.init()

for i in range( pygame.midi.get_count() ):
        r = pygame.midi.get_device_info(i)
        (interf, name, input, output, opened) = r

        in_out = ""
        if input:
            in_out = "(input)"
            if name == "microKEY-37":
                keyboardID = i
                print "KEYBOARD:",i
            if name == "2- MIDISPORT 2x2 In A":
```

```
                pedalID = i
                print "PEDAL:",i
        if output:
            in_out = "(output)"

##        print ("%2i: interface :%s:, name :%s:, opened :%s:  %s" %
##               (i, interf, name, opened, in_out))

Keyboard = pygame.midi.Input( keyboardID )
Pedal = pygame.midi.Input( pedalID )

pygame.display.set_mode((1,1))


maneuver = 0
pitch = 0
zeta = 0
omega = 0

TCP_IP = '127.0.0.1'
TCP_PORT = 5005
BUFFER_SIZE = 6

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))

## output
instrument = 0 # GRAND PIANO
midi_out = pygame.midi.Output(pygame.midi.get_default_output_id(), 0)
midi_out.set_instrument(instrument)


going = True
while going:
    events = event_get()
    for e in events:
        if e.type in [pygame.midi.MIDIIN]:
            print (e)
            print (e.data1)
```

```
    if Keyboard.poll():
        keyb_events = Keyboard.read(10)
        keyboard_events = pygame.midi.midis2events(keyb_events, Keyboard.device_id)

        for k_e in keyboard_events:
            if k_e.status == 144:
                MIDInotesON[k_e.data1] = 1
                msg = "p" + format(k_e.data1 + 50, '05')
                s.send(msg)
                midi_out.note_on(k_e.data1,127)
            if k_e.status == 128:
                midi_out.note_off(k_e.data1)
                MIDInotesON[k_e.data1] = 0
            event_post( k_e )

    if Pedal.poll():
        ped_events = Pedal.read(10)
        pedal_events = pygame.midi.midis2events(ped_events, Pedal.device_id)
        for p_e in pedal_events:
            if p_e.status == 176:
                if p_e.data1 == 27: ## left pedal
                    msg = "d" + format(p_e.data2*1.2/127.0 + 0.2 + 50, '05')
                    s.send(msg)
                if p_e.data1 == 7: ## right pedal
                    msg = "w" + format(p_e.data2*5/127.0 + 0.5 + 50, '05')
                    s.send(msg)
            if p_e.status == 192: ## right pedal
                    msg = "m" + format(p_e.data1-39 + 50, '05')
                    s.send(msg)
            event_post(p_e)


        print msg

del Keyboard
del Pedal
s.close()
pygame.midi.quit()
```

Appendix J

model.py

```
## quadrotor parameters
Ixx = 0.0021
Iyy = 0.0018
Izz = 0.0027
JTP = 0
l = 0.19 #m
mass = 0.431 #g
g = 9.81 #N/m/s
dt = 0.01
kYaw = 0.1
```

Bibliography

[1] News Article *More About Balloons* in Scientific American, 1849, available online at http://www.ctie.monash.edu/hargrave/rpav_home.html#Beginnings

[2] A. Puri, A Survey of Unmanned Aerial Vehicles (UAV) for Traffic Surveillance

[3] D. W. Casbeer, R. W. Beard, T. W. McLain, S. Li, R K. Mehra, *Forest Fire Monitoring With Multiple Small UAVs*, American COntrol Conference, June 8-10 2005

[4] F. Caballero, L. Merino, J. Ferruz, A. Ollero, *A visual odometer without 3D reconstruction for aerial vehicles. Applications to building inspections*, Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, April 2005.

[5] J. R. MArtinez-De Dios, A. Ollero, *Automatic Detection of Windows Thermal Heat Losses in BUildings using UAVs*, World Automation Congraess (WAC), Budapest, Hungary, July 24-26 2006.

[6] Amazon.com, *Amazon Prime Air*, available on http://www.amazon.com/primeair

[7] Fischer, J, *CES 2015: Drones, Drones, Drones*, pcmag.com, available online at http://www.pcmag.com/article2/0,2817,2474885,00.asp

[8] Hartmann, WM, *Signals, Sound, and Sensation*, pp. 145, 284, 287, Springer.

[9] A. Schollig, F. Augugliaro, S. Lupashin and R. D'Andrea, *Synchronizing the Motion of a Quadrocopter to Music*, in IEEE International Conference on Robotics and Automation, Anchorage, Alaska, USA, 2010.

[10] A. P. Schoellig, C. Wiltsche and R. D'Andrea, *Feed-Forward Parameter Identification for Precise Periodic Quadrocopter Motions*, in Proc. of the 2012 American Control Conference (ACC), 2012.

[11] M.M. Wanderley, P. Depalle, O. Warusfel, *Improving instrumental sound synthesis by modeling the effects of performer gesture*, in ICRC, 1999

[12] V. Kumar, *Robot Quadrotors Perform James Bond Theme*, University of Pennsylvania, General Robotics, Automation, Sensing and Perception (GRASP) Lab, Feb 2012, available at http://www.youtube.com/watch?v=_sUeGC-8dyk

[13] C. Cadoz, M. M. Wanderley, *Gesture - Music*, Trends in Gestural Control of Music, IRCAM 2000

[14] J. van der Linden, Erwin Schoonderwaldt, J. Bird, R. Johnson, *MusicJacket Combining Motion Capture and Vibrotactile Feedback to Teach Violin Bowing*, in IEEE Transactions on instrumentation and measurement, Vol. 60, No. 1, January 2011

[15] I. Choi, *Cognitive Engineering of Gestural Primitives for Multi-modal interaction in a Virtual Environment*, IEEE International Conference on Systems, Man, and Cybernetics (Vol. 2), San Diego, CA, 11-14 Oct 1998, pp 1101-1106

[16] G. Kurtenbach, E. A. Hulteen, *Gestures in Human-Computer Interaction*, In B. Laurel (ed.): The Art of Human-Computer Interaction, Reading, Mass.: Addison-Wesley, 1990, page 310

[17] P. Viviani, *Pleins et dlis*, Science et Vie, numro spcial, Le cerveau et le mouvement. pp. 36-47, 1998

[18] K. Murata, K. Nakadai, R. Takeda, H. G. Okuno, T. Torii, Y. Hasegawa and H. Tsujino, *A Beat-Tracking Robot for Human-Robot Interaction and Its Evaluation*, in IEEE-RAS International Conference on Humanoid Robots, Daejeon, Korea, 2008.

[19] K. Murata, K. Nakadai, K. Yoshii, R. Takeda, T. Torii, H. G. Okuno, Y. Hasegawa and H. Tsujino, *A Robot Uses Its Own Microphone to Synchronize Its Steps to Musical Beats While Scatting and Singing*, in IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 2008.

[20] D. M. Randel, *The Harvard Dictionary of Music*, Harvard University Press Reference Library, Dec 28, 2003

[21] E. W. Large, J. F. Kollen, *Resonnance and the Perception of Musical Meter*, Connection Science, Vol. 6, Ns. 2 & 3, 1994

[22] L. P. Rabiner, *On the Use of Autocorrelation Analysis for Pitch Detection*, IEEE Transactions on Acoustics, Speech, and Signal processing, Vol. ASSP-25, No. 1, February 1977, pp 24-33

[23] P. de la Cuadra, A. Master, C. Sapp, *Efficient Pitch Detection Techniques for Interactive Music*, Center for Computer research in Music and Acoustics, Stanford University

[24] E. D. Scheirer, *Tempo and beat analysis of acoustic musical signals*, Journal of Acoustic Society of America, January 1998

[25] P. Sofras, *Dance Composition Basics: Capturing the Choreographer's Craft*, Human Kinetics, 2006

[26] J. M. Smith-Autard, *Dance Composition: A Practical Guide to Creative Success in Dance Making*, Bloomsbury Methuen Drama; 6th Revised edition edition (Aug. 3 2010)

[27] M. Goto, Y. Muraoka, *An Audio-based Real-time Beat Tracking System and Its Applications*, in Proceedings of the 2001 International Computer Music Conference (ICMC'1998), 1998.

[28] O. Rogalla, M. Ehrenmann, R. Zollner, R.Becher, R.Dillmann, *Using Gesture and Speech Control for Commanding a Robot Assistant*, Proceedings of the 2002 IEEE International Workshop on Robot and Human Interactive Communication, Berlin, Germany, Sep 25-27 2002

[29] Cirque du Soleil, ETH Zurich, and Verity Studios, *SPARKED: A Live Interaction Between Humans and Quadcopters*, available online at https://www.youtube.com/watch?v=6C8OJsHfmpI, published on Sep 22, 2014

[30] D. Perzanowski, A. C. Schultz, W. Adams, E. Marsh, M. Bugajska, *Building a Multimodal Human-Robot Interface*, IEEE Intelligent Systems, January/February 2001

[31] L. R. Rabiner, M. J. Cheng, A. E. Rosenberg and C. A. McGonegal, *A Comparative Performance Study of Several Pitch Detection Algorithms*, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vols. ASSP-24, no. No. 5, pp. 399-418, 1976.

[32] W. B. Kuhn, *A Real-Time Pitch Recognition Algorithm for Music Applications*, Computer Music Journal, Vol. 14 No. 3, Autumn 1990, pp. 60-71

[33] S. Harbeck, A. Kiebetaling, R. Kompe, H. Niemann and E. No"th, *Robust Pitch Period Detection Using Dynamic Programming with an ANN Cost Function*, Univ. Erlangen-Nurnberg, Lehrstuhl fur Mustererkennung (Inf. 5), Martensst r. 3,91058 Erlangen, F.R. of Germany, 1995.

[34] E. Benetos and S. Dixon, *Joint Multi-Pitch Detection Using Harmonic Envelope Estimation for Polyphonic Music Transcription*, IEEE Journal of Selected Topics in Signal Processing, vol. 5, no. 6, pp. 1111-1123, 2011.

[35] E. Benetos and S. Dixon, *A Temporally-Constrained Convolutive Probabilistic Model for Pitch Detection*, in IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, 2011.

[36] H. Ding, B. Qian, Y. Li, Z. Tang, *A method combining LPC-Based Cepstrum and Harmonic Product Spectrum for Pitch Detection*, Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2006

[37] D. G. Childers, D. P. Skinner, R. C. Kemerait, *The Cepstrum: A Guide to Processing*, Proceedings of the IEEE

[38] A. V. Oppenheim, R. W. Schafer, *From Frequency to Quefrency: A History of the Cepstrum*, IEEE Signal Processing Magazine, September 2004.

[39] M. R. Schroeder, *Period Histogram and Product Spectrum: New Methods for Fundamental Frequency Measurement*, Acoustical Society of America, 1968

[40] J. P. Kroeker, *Resonant aperture detection of natural resonances*, Wave Reactive Network, available online at https://drive.google.com/file/d/0B-XL4upneCeCdzhGNXJLNEtaMjQ/view

[41] A. V. Oppenheim, R. W. Schafer, *Digital Signal Processing*, Prentice Hall, 1975

[42] A. V. Oppenheim, R. W. Schafer, *Discrete-time signal processing*, Second edition, Prentice Hall, 1998

[43] V. Fromkin, J. Rodman, *An Introduction to Language*, CBS College Publishing, 1983

[44] F. Weichert, D. Bachmann, B. Rudak, D. Fisseler, *Analysis of the Accuracy and Robustness of the Leap Motion Controller*, in Sensors 2013, pp 6380-6393

[45] Z. Ren, J. Meng, J. Yuan, Z. Zhang, *Robust hand gesture recognition with kinect sensor*, In Proceedings of the 19th ACM international conference on Multimedia (MM '11). ACM, New York, NY, USA, 759-760.

[46] K. K. Biswas, S. K. Basu, *Gesture recognition using Microsoft Kinect*, 5th International Conference on Automation, Robotics and Applications (ICARA),pp 100-103, Dec 2011

[47] O. Patsadu, C. Nukoolkit, B. Watanapa, *Human gesture recognition using Kinect camera*, in International Joint Conference on Computer Science and Software Engineering (JCSSE), pp 28-32, May 30 - June 1 2012

[48] T. Schlmer, B. Poppinga, N. Henze, S. Boll. *Gesture recognition with a Wii controller*, In Proceedings of the 2nd international conference on Tangible and embedded interaction (TEI '08). ACM, New York, NY, USA, pp 11-14.

[49] Thalmic Labs, *Myo - Gesture control armband by Thalmic Labs*, available online at https://www.thalmic.com/en/myo/

[50] A. Malima, E. Ozgur, M. Cetin, *A Fast Algorithm for Vision-Based Hand Gesture Recognition for Robot Control*, in IEEE Signal Processing and Communications Applications, Antalya, Turkey, 2006

[51] Y. Wu, T. S. Huang, *Vision-Based Gesture Recognition: A Review*, in Gesture-Based Communication in Human-Computer Interaction, International GestureWorkshop, GW99 Gif-sur-Yvette, France, March 17-19, 1999 Proceedings pp 103-115

[52] L. Dipietro, A. M. Sabatini, P. Dario, *A Survey of Glove-Based Systems and Their Applications*, in IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews, Vol 38, No. 4, July 2008

[53] J. P. Wachs, M. Kolsch, H. Stern, Y. Edan, *Vision-Based Hand-Gesture Applications*, in Communications of the ACM, February 2011, Vol 54, No. 2, pp 60-71

[54] F. Song, W. B. Croft, *A General Language Model for Information Retrieval*, in Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pp 279-280

[55] Y. Lv, C. Zhai, *Positional Language Models for Information Retrieval*, in SIGIR '09, Boston, MA, Jul 19-23 2009

[56] L. R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, Vol 77, No. 2, Feb 1989

[57] D. Hakkani-Tur, G. Riccardi, A. Gorin, *Active Learning for Automatic Speech Recognition* in Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on (Vol. 4), Orlando, FL, 13-17 May 2002, pp 3904-3907

[58] M. J. F. Gales, S. J. Young, *Robust Continuous Speech Recognition Using Parallel Model Combination*, IEEE transactions on Speech and Audio Processing (Vol. 4, Issue 5), pp 352-359, 1996

[59] J. Bachus, *The Acoustical Foundations of Music*, W W Norton & Co Inc (Np); 2nd Revised edition edition (Nov. 17 1977)

[60] J. W. Cooley, J. W. Tukey, *An Algorithm for the Machine Calculation of Complex Fourier Series*, Mathematics of Computation, Vol. 19, No. 90, Apr 1965, pp 297-301

[61] L. Robles, M. A. Ruggero, *Mechanics of the Mammalian Cochlea*, Physiological Reviews Published 1 July 2001 Vol. 81 no. 3, pp. 1305-1352

[62] Peter van Hengel, *Cochlea (conceptual documentation)*, available online at http://www.ai.rug.nl/acg/cpsp/docs/cochleaModel.html

[63] Y. Gong, *Speech recognition in noisy environments: A survey*, in Speech Communication (Vol. 16, Issue 3), April 1995, Pages 261291

[64] R. B. Dannenberg, B. Brown, G. Zeglin, R. Lupish, *McBlare: A Robotic Bagpipe Player*, in Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05), Vancouver BC Canada, pp 80-84

[65] G. Hoffman, G. Weinberg, *Shimon: An Interactive Improvisational Robotic Marimba Plauer*, in Computer-Human Interaction, Atlanta, GA, April 10-15 2010

[66] A. Alford, S. Northrup, K. Kawamura, K.W. Chan, J. Barile, *Music Playing Robot*, in Proceedings of the Conference on Field and Service Robots, pp. 29-31. 1999.

[67] Y. Kusuda, *Toyota's violin playing robot*, Industrial Robot: An International Journal, Vol. 35 Iss 6 pp. 504 - 506

[68] D. Zhang, J. Lei, B. Li, D. Lau, C. Cameron, *Design and analysis of a piano playing robot*, in International Conference on Information and Automation (ICIA), pp 757-761, 2009

[69] J. Santana, X. Smith, Saatchi P, *MEET YOUR CREATOR - QUADROTOR SHOW*, available online at https://www.youtube.com/watch?v=cseTX_rW3uM

[70] MIDI Manufacturers Association, *The MIDI standard*

[71] E. R. Miranda, M. M. Wanderley, *New Digital Musical Instruments: Control and Interaction beyond the Keyboard*, The Computer Music and Digital Audio Series, Vol. 21

[72] S. Iba, J. M. V. Weghe, C. J. J. Paredis, P. K Khosla, *An Architecture for Gesture-Based Control of Mobile Robots*, in Proceedings of the IEE/RSJ International Conference on Intelligent Robots and Systems, 1999

[73] S. Waldherr, R. Romero, S. Thrun, *A Gesture Based Interface for Human-Robot interaction*, in Autonomous Robots 9, pp 151-174, 2000

[74] D. Bohus, C. W. Saw, E. Horvitz, *Directions Robot: In-the-Wild Experiences and Lessons Learned*, in Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014) Paris, France, May 5-9, 2014

[75] M. K. Apostolos, M. Littman, S. Lane, D. Handelman, J. Gelfand, *Robot Choreography: An Artistic-Scientific Connection*, in Computers Math. Applic. Vol 32, No. 1, pp 1-4, 1996

[76] S.F. de Sousa Junior, M.F.M. Campos, *Shall we dance? A music-driven approach for mobile robots choreography*, Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on , vol., no., pp.1974,1979, 25-30 Sept. 2011

[77] T. Luukkonen, *Modelling and control of quadcopter*, Aalto University, School of Science, Independent research project in applied mathematics, August 22, 2011, available online at http://sal.aalto.fi/publications/pdf-files/eluu11_public.pdf

[78] S. Bouabdallah, R. Siegwart, *Full Control of a Quadrotor*, Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems San Diego, CA, USA, Oct 29 - Nov 2, 2007

[79] S. Bouabdallah, A. Noth, R. Siegwan *PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor*, in Proceedings of 2004 1EEElRS.J Internationel Conference On Intelligent Robots and Systems, Sep 28 - Oct 2, 2004, Sendal, Japan

[80] D. Tymoczko, *A geometry of Music: Harmony and Counterpoint in the Extended Common Practice*, Oxford University Press, 2011.

[81] D. M. Huber, *The MIDI Manual: A Practical Guide to MIDI in the Project Studio*, Focal Press, 3rd Edition, 2007.

[82] R. George, 2012, *Project 326: MIDI Merlin* [Online]. Available: http://cycling74.com/project/midi-merlin/

[83] Music Technology Group of Universitat Pompeu Fabra, 2012, *Algorithm reference: PredominantMelody - Essentia 2.0.1 documentation* [Online]. Available: http://essentia.upf.edu/documentation/reference/streaming_PredominantMelody.html

[84] S. Dixon, *An Interactive Beat Tracking and Visualisation System*, in Proceedings of the 2001 International Computer Music Conference (ICMC'2001), 2001. [Online] Available: http://code.soundsoftware.ac.uk/projects/beatroot

[85] Benward & Saker, *Music: In Theory and Practice, Vol. I*, p. 67 & 359. Seventh Edition.

[86] MohammedAG, *[MOD][Xposed] Google Search / Now API*, available online at http://forum.xda-developers.com/xposed/modules/mod-google-search-api-t2554173

[87] Wave Reactive, available online at http://www.wavereactive.com/

[88] Circular Logic, *Circular Logic — InTime^{TM}- Realtime synchonization, beat tracking and tempo mapping*, [Online] Available: http://www.circular-logic.com/products.html

[89] I. Hattwick, J. Malloch, M. Wanderley, *Forming Shapes to Bodies: Design for Manufacturing in the Prosthetic Instruments*, NIME, Goldsmiths College, Lewisham Way, London, United Kingdom 2014

[90] N. J. Conard, M. Malina, S. C. Munzel, *New flutes document the earliest musical tradition in southwestern Germany*, Nature, Aug 6th 2009 available online at http://www.nature.com/nature/journal/v460/n7256/pdf/nature08169.pdf

[91] J. Malloch, *Adding Vibrotactile Feedback to the T-stick Digital Musical Instrument*, report submitted in partial fulfillment of course ECSE-618 "Haptics", McGill University, Fall 2007, available online at http://www.vigliensoni.com/McGill/CURSOS/2009_09/MUMT620/PRESENTATION/Presentatio
%20Actuators/malloch_vibrotactile.pdf

[92] J. Blacking, *How musical is man?*, 1973

[93] Behringer, *Behringer: FCB1010*, available online at http://www.behringer.com/EN/Products/fcb1010.aspx

[94] Toddatkins, *Managing emotions - Identifying feelings*, available online at http://batonrougecounseling.net/managing-emotions/, 12 February 2011

[95] I. Andjelkovic, M. Hetrick, F. M. Scotio, *BrainTag*, avilable online at http://www.mat.ucsb.edu/ ivana/200a/background.htm

[96] S. Lupashin, A. Schollig, M. Sherback, R. D'Andrea, *A simple learning strategy for high-speed quadrocopter multi-flips*, in Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp 1642-1648

[97] A. P. Schoellig, H. Siegel, F. Augugliaro, R. DAndrea, *So You Think You Can Dance? Rhythmic Flight Performances with Quadrocopters*, Controls and Art, Springer International Publishing Switzerland 2014, pp 73-105

[98] *Max is a visual programming language for media*, available online at https://cycling74.com/products/max/

[99] zplane, *zplane – music processing and analysis technology - [aufTAKT] tempo and beat tracking*, [Online] Available: http://www.zplane.de/index.php?page=description-auftakt-ma

[100] T. Bresciani, *Modelling, Identification and Control of a Quadrotor Helicopter* M.S. thesis, Dept. Automatic Control, Lund Univ., Lund, Sweden, 2008.

[101] N. Michael et al. *"The GRASP Multiple Micro-UAV Test Bed IEEE Robotics & Automation Magazine*, pp. 56-65, Sept. 2010.

[102] Y. E. Kim, E. Schmidt, L. Emelle, *MoodSwings: A Collaborative Game for Music Mood Label Collection*, Session 2c - Knowledge Representation, Tags, Metadata, ISMIR 2008, pp 231-236

[103] B. L. Wheeler, *Relationship of Personal Characteristics to Mood and Enjoyment after Hearing Live and Recorded Music and to Musical Taste*, from the SAGE Social Science Collections

[104] P. Kenealy, *Validation of a music mood induction procedure: Some preliminary findings*, in Cognition and Emotion, 2:1, 41-48

[105] C. Laurier, J. Grivolla, P. Herrera, *Multimodal Music Mood Classification using Audio and Lyrics*, International Conference on Machine Learning and Applications, 2008

[106] C. Laurier, M. Sordo, J. Serra, P. Herrera, *Music Mood Representations from Social Tags*, in 10th International Society for Music Information Retrieval Conference (ISMIR 2009), pp 381-386

[107] MIREX, *Audio Music Mood Classification*, available online at http://www.music-ir.org/mirex/wiki/2010:Audio_Music_Mood_Classification

[108] M. Di Perna, M. El-Jiz, C. Glass, P. Daugherty, L. Rodrigues, "Spiri - a robot in sync with music", Concordia University, January 2014, available at https://www.youtube.com/watch?v=kJSraZdnd3Y.

[109] C. Ossa-Gomez, *"Design, Construction and Control of a Quadrotor Helicopter Using a New Multirate Technique"*, Thesis, Concordia University, Montreal QC, June 2012.

[110] S. Lupashin, M. Hehn, M.W. Mueller, A.P. Schoellig, M. Sherback, R. DAndrea, *A platform for aerial robotics research and demonstration: The Flying Machine Arena"*, Mechatronics, Volume 24, Issue 1, 2014, pp. 4154.

[111] G.F. Franklin, J. David Powell, A. Emami-Naeini, *Feedback Control Of Dynamic Systems*, Pearson Prentice Hall, Fifth Edition, Chapter 7.

[112] B. Etkin, L. Duff Reid, *Dynamics of Flight, Stability and Control*, McGraw-Hill, Third Edition, 1996.

[113] William L. Brogan, "Design of Linear Feedback Control Systems" in *Modern Control Theory*, 3rd Ed. New Jersey: Prentice-Hall, 1991

[114] M. R. Jardin, E. R. Mueller, *Optimized Measurements of Unmanned-Air-Vehicle Mass Moment of Inertia with a Bifilar Pendulum*, in Journal of Aircraft, Vol. 46, No. 3, May-June 2009

[115] KORG, *microKEY USB-POWERED KEYBOARD - MIDI CONTROLLERS - KORG*, available online at http://www.korg.com/us/products/controllers/microkey/

[116] Python Software Foundation, *What is Python? Executive Summary - Python.org*, available online at https://www.python.org/doc/essays/blurb/

[117] Visual Python, *VPython*, available online at http://vpython.org/

[118] Pygame, *Wiki*, available online at http://www.pygame.org/wiki/about

[119] Pygame, *pygame.midi - Pygame v1.9.2 documentation*, available online at https://www.pygame.org/docs/ref/midi.html

[120] . Python Software Foundation, *9.2. math - Mathematical functions - Python 2.7.10 documentation*, available online at https://docs.python.org/2/library/math.html

[121] Berry, Wallace (1976/1986), *Structural Functions in Music*, p.349.

[122] C. Glass, *Quadrotor choreography performance by Caroline Glass*, available online at https://youtu.be/wVw2M_RSegg

[123] Ableton, *Music production with Live 9 and Push - Ableton*, available online at https://www.ableton.com/

[124] nerds.de, *ipMIDI - MIDI over Ethernet port*, available online at http://www.nerds.de/en/ipmidi.html