6.111 Project Proposal

Real-time Auto-Tune

Ishwarya Ananthabhotla , Trevor Walker

1 Background

In today's contemporary music produced by vocal artists, live recordings are almost necessarily subjected to improvement or correction by means of a process known as auto-tuning. Auto-tuning is a process that involves taking individual notes that are sung by a performer and digitally transposing them by minor increments to match the series of notes originally intended to form the melody. This type of system typically exists in the form of software that can be used to enhance vocals after live recordings have been taken, and is usually operated by recording engineers in professional studios. On an abstracted level, these tools operate by classifying and altering each recorded note as required in the frequency domain to match (or at least come closer to) the intended note in the song.

2 Motivation

In this particular project, we aim to design, implement, and test a system that is capable of performing auto-tuning in real-time. While signal processing, computation, and shifting in the frequency domain are concepts that have traditionally been approached from a software perspective, this project will focus on a complete hardware implementation. As a result, we seek to gain a better understanding of the design decisions and trade-offs required to build a comparatively robust hardware system. We are also hoping to use this project as an opportunity to explore the realm of digital audio processing, a field in which both student authors have an active interest. Finally, we would like to implement some additional features that are non-traditional in an auto-tuning system, and evaluate

the output sound quality of the reproduced audio in response to these added features from a qualitative standpoint.

3 System Overview

On a high level, our system can be represented by the system diagram shown in Figure 1.

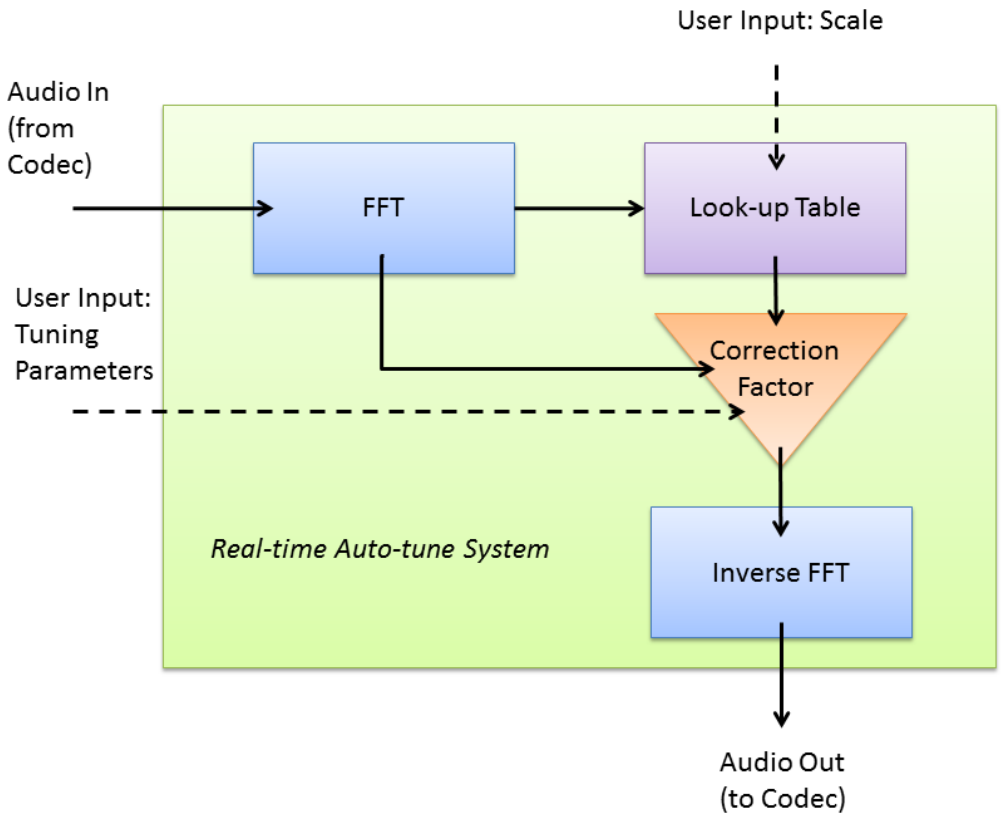

Figure 1: Block diagram of our auto-tune system

The core of the system will be restricted to one or more Xilinx FPGA boards, along with the use of a few minor peripheral devices (such as microphones and speakers) for the purpose of capturing and delivering audio. As shown in Fig. 1, audio will be input into the system via a microphone, filtered, sampled, digitized by the AC97 Codec with which we have already experimented in this class. Then, for a pre-determined window of data points, we will use the CoreGen module to perform an FFT on this data in order to obtain a spectrum of frequency distribution for that particular sample. The system will

identify the fundamental frequency from this spectrum, and compare it to an "a priori" spectrum which represents the allowable frequencies for this particular sample (where a series of such samples form the song). This spectrum will be generated offline by a user specifying which notes are allowed to belong in this song. The system will then compare the song data spectrum with the a priori spectrum to identify the note in the latter which is closest to the one that has been sung, and the degree of correction that is required. This is portion is executed via a lookup table. The entire spectrum of the song data will then be shifted by this degree, a parameter which can be tuned slightly by a user to influence the amount of naturalness that is preserved in the output. Additionally, the user will have the opportunity to tune the rate of correction, another factor which will determine the naturalness of the sound output. Finally, the system will perform the inverse of FFT of this newly shifted spectrum, outputting the time-domain waveform back through the Codec and out to speakers or headphones that will be connected to the board.

4 Modules

4.1 FFT

The FFT, or Fast Fourier Transform module, is the most crucial part of the system. Its responsibility is to take the discrete Fourier transform of incoming audio samples, to transform them from the time domain to the frequency domain. There are severial design tradeoffs to be considered in the design of this system's FFT - a larger sample increases the system's ability to distinguish adjacent pitches, but requires more resources on the FPGA. Because pitch is a logarithmic function of frequency, the difference in frequency between adjacent pitches increases with frequency, so higher pitches are easier to distinguish from each other. The pitch discrimination capability of the FFT, therefore, determines the lowest pitches that the system will be able to accurately identify. The size of the module can also be decreased in exchange for an iterative structure, though this introduces greater latency and lowers the module's throughput. Because the FFT is complex to design in Verilog, we will use a generator like Xilinx's CoreGen to produce the Verilog source code. Ishwarya will implement this module.

4.2 LUT

The lookup-table (LUT) module is likewise essential to the system's operation. For the base-level ("B"-level) system's implementation (pitch detection and display), the structure of the lookup table is relatively simple, and its function is simply to map frequency to a musical note. For example, a detected frequency of 440Hz corresponds to the note A_4 (the A above middle C) in this table.

In the case of the higher-level ("A"-level) implementation, the table is more complex. Its index is still the detected fundamental frequency, but its outputs are now given by $LUT[i] = f_c/f_i$, where $LUT[i]$ represents the LUT value for the FFT result at i, f_c represents the closest frequency to the detected frequency, and f_i represents the actual frequency of the FFT sample at i. This table is generated at startup based on the user-selected set of permissible pitches. The output of this table is the ratio between the desired pitch and the pitch the user has sung; therefore, it is the factor that the FFT result must be multiplied by to bring the user's singing to the correct note.

Finally, for the most difficult level of implementation, the table is generated anew with every sample based on the changing set of permissible pitches. Trevor will implement this module.

4.2.1 Peak Finder

Between the FFT and the look-up table itself, it is necessary to determine the fundamental frequency being sung; this is accomplished by iterating over the result of the Fourier transform and determining the component with the largest amplitude. It is the frequency of this component that is used as the index of the LUT.

4.3 Pitch Display

This component is fairly simple; it consists of capturing the output of the (simpler) LUT, and converting that information into a display for the user. This display will be implemented on the labkit's 5x7 LED array with a version of the 6.111 hex_display module modified to display the letters from A to G,♯, and ♭. Trevor will implement this module.

4.4 Corrector

The function of the corrector module is to perform the appropriate shift on the FFT result, as determined by the LUT. This is accomplished by multiplying the FFT result by

the pitch correction factor from the LUT. Additionally, the user controls two parameters affecting this component: the degree of correction and the rate of correction. The degree of correction ranges from 0 to 1 (scaled to an integer for compatibility with the FPGA's fixed-point computation), by which the pitch correction factor is multiplied before the correction is applies. The correction rate parameter allows the user to control how natural the corrected audio sounds by controlling how aggressively the correction is applied - the slower the rate of correction, the more natural the sounds. This control will be implemented by the inclusion of memories in this module, to store the current correction factor and allow it to change over time as a sung note continues. Trevor will implement this module.

4.5 IFFT

The inverse Fast Fourier Transform will be very similar to the forward transform described previously; it will have the same value for N, and many of the calculations necessary are the same. All the same trade-offs apply too, and it is likely that the parameters for this module will be the same as those for the FFT. The function of this module is to transform the frequency-domain sound representation produced by the FFT and modified by the corrector to a time-domain representation suitable for loading into the AC97 codec and played on the speakers. Ishwarya will implement this module.

5 Extra Features

For the "A+"-level implementation, the system gains an addditional feature which was briefly discussed in the LUT section: the ability for the user to provide a sequence of pitches in real time and have those pitches used as the target pitch for correction. This mode would allow a user to indicate the notes that they should be singing, and will permit interesting effects such as warping pitch to target notes very far from what is actually being sung. It will also eliminate the potential for harsh jumps between frequencies while singing, as only one note will be targeted at a time.