KUNGL TEKNISKA HÖGSKOLAN

Department of Speech, Music and Hearing

Analysis and Simulation of Ensemble Sounds
M.Sc. thesis

Daniel Kahlin

24th March 2000

Examensarbete för civilingenjörsexamen
Institutionen för tal, musik och hörsel
Handledare Sten Ternström
Examinator Johan Sundberg

Written by Daniel Kahlin.

`$Id: report.tex,v 1.74 2000/03/24 15:11:56 tlr Exp $`

Abstract

Analysis and Simulation of Ensemble Sounds

The so-called "chorus" or "ensemble" effect is interesting both musically and perceptually. It is usually imitated in effect devices using slowly varying time shifts, giving the impression of rotating speakers rather than that of an ensemble. Dolson found in 1983 that the quasi-random amplitude modulation of beating partials alone can cue the perception of ensemble. The small changes in frequency, he found, are less salient perceptually. This suggests an alternative simulation of the chorus effect.

Attempts were made to corroborate Dolson's finding, and to simulate ensembles in the frequency domain by modulating only partial tone amplitudes, using three approaches: filter banks, real-valued FFT:s and complex-valued FFT:s. The exact partial envelopes of a choral sound were found to be elusive, partly because the sidebands of one partial will overlap its neighbours at higher frequencies. The methods used are described, and the outcome of the trials is discussed.

Sammanfattning

Analys och Simulering av Ensembleljud

Den så kallade "korus" eller "ensemble" effekten är intressant både musikaliskt och perceptionsmässigt. I effektenheter imiteras den vanligen med hjälp av långsamt förändrade tidfördröjningar, vilket snarare ger intrycket av roterande högtalare, än av en ensemble. Dolson upptäckte 1983 att den kvasislumpmässiga amplitudmodulationen som uppstår när deltoner svävar mot varandra i sig räcker för att skapa en känsla av ensemble. Han fann att de små förändringarna i frekvens är mindre tydliga perceptionsmässigt. Detta ger ett uppslag till en alternativ simulering av korus-effekten.

Försök gjordes att bekräfta Dolsons upptäckt, och att simulera ensembler i frekvensdomänen genom att endast modulera deltonernas amplituder. Detta gjordes på tre sätt: filterbanker, reellvärda FFT:er och komplexvärda FFT:er. Deltonernas exakta amplitudenveloper i ett körljud visade sig vara komplicerade, delvis på grund av att sidbanden kring en delton kommer att överlappa sina grannar vid höga frekvenser. De använda metoderna beskrivs, och resultaten av försöken diskuteras.

A concise version of this report, with an extended version of the filter-bank model, was presented at Euromicro 99 in Milano, Italy, Sept 8-10 1999.[8]

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Why would we want a new method for making ensemble sounds? A device that could simulate the sound of an ensemble from one single voice would be of interest to many musicians and recording studios. A common technique for achieving ensemble sound is to record multiple versions of the same sound on parallel tracks and then mix them together, so-called overdubbing. This is however time consuming, and requires a "live" sound source to begin with. Repeatedly adding the same sequenced synthesizer track will only produce increased loudness and maybe some comb filter effects. Adding multiple takes of a guitarist or singer together works well, apparently because of inevitable small differences between takes.

A standard device for simulating an ensemble is the so-called chorus effect, which with small variations can be found in most commercial synthesizers and effect processors. The conventional chorus device takes a somewhat heuristic approach, in which three versions of the input signal are subjected to varying delays and then summed together. This chorus effect is popular in pop and rock music, but it does not sound quite like a choir. It is a rather basic approximation of what happens when three voices sound together. This algorithm uses only time delays which are easy to implement, but perhaps too simple for obtaining a realistic ensemble sound.

The purpose of the present work was to investigate whether an algorithm can be devised that can transform one voice into a large ensemble of voices, such as a choir or a string section. Ideally, the algorithm should also be implementable in a real time device.

1.2 Earlier work

Descriptions of practical implementations of the so called chorus effect are mostly found in patent applications, e.g., Adachi [2], Cotton [3]. Typically three versions of the input signal with are summed together with different time varying delays. The delays are varied in time according to a function which mostly seems to be arrived at by trial and error. A common modulation signal consists of a low frequency sinewave of about 0.7-0.8 Hz with a smaller ampli-

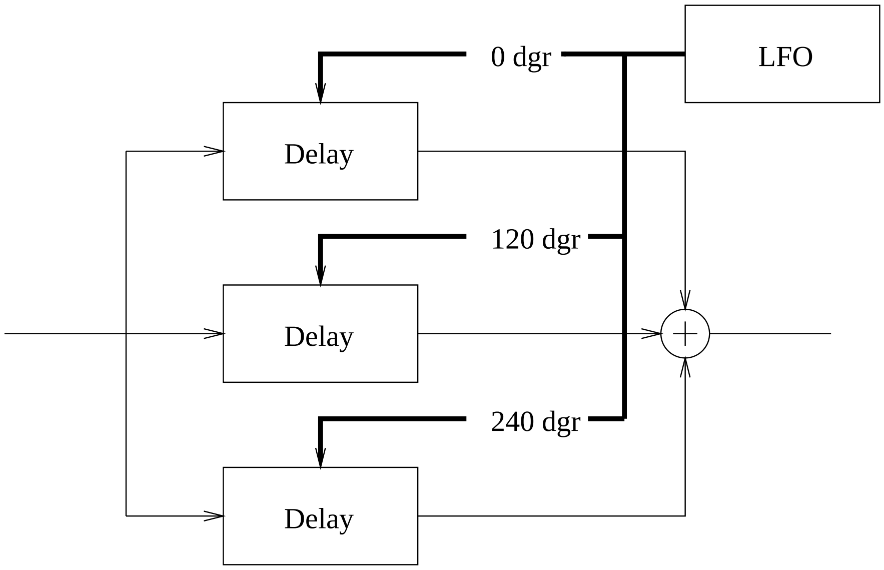


Figure 1.1: Principle of the conventional chorus effect

tude sinewave of 8-10 times that frequency superimposed. This signal is then phase shifted to obtain three different signals, 0, 120 and 240 degrees. "see figure 1.1 on page 2." The delay time varies between about 10 to 40 ms. It is important that the higher frequency modulation component affect delay time equally during the period of the lower frequency component, i.e. the relationship between the modulation signal and the delay time must be linear. This becomes an issue mostly in analog implementations. The result is a sense of ensemble, especially if this is applied in stereo with slightly different parameters for the left and right channels.

Dolson [1] made experiments on solo and ensemble violin sounds. He reports that violin sounds are in fact entirely harmonic, except for the attack portion. He found that the sustain portion alone is sufficient for obtaining ensemble sensation. He also found that at least 4-8 partials are necessary for achieving ensemble sensation. He verified that if the frequency variation of the harmonic is Δ_f, then the frequency variation of the n:th harmonic is $n\Delta_f$, as expected. The brain appears to recognize this pattern; Dolson states that beats in the low end of the spectrum must be slow, while beats in the high end of the spectrum must be rapid for the ear to accept the sound as coming from an ensemble. In fact Dolson found that such amplitude modulations appear to be sufficient. He also tried simulating using only frequency modulation, which gave a weaker sense of ensemble.

These findings suggest that it might be possible to simulate an ensemble using some frequency domain transform to manipulate only the amplitudes of frequency bands or the individual partials. This would have the advantage of not requiring frequency variations in the source signal, when synthesizing an ensemble.

Thus, in these experiments we wished to explore whether ensemble effects might be obtained by modulating signals in the frequency domain rather than in the time domain. An additional constraint is that the chosen algorithms should theoretically be implementable in a real time device, i.e no long latencies may be present.

1.3 Ensemble sounds

The type of ensemble sounds addressed in this report is represented by a group of male singers singing the same vowel in unison, that is at approximately the same pitch. Typically the amplitude envelope of such an ensemble sound has little modulation, whereas each partial by itself has practically 100% modulation. For our purposes we will assume that the singers of an ensemble are independent. By this we mean that if the singers were recorded one at a time, the sum of these recordings would still sound like the same ensemble, at least for sustained tones. It could be argued that a good ensemble sound requires singer interaction, in other words, that the singers are not independent. This may, or may not be the case, but the assumption will hopefully be sufficient for our task.

Chapter 2

Methods and Results

2.1 Approaches

Three different methods of frequency domain analysis and modulation were tried, with progressively higher demands for precision and information content.

- filter bank - accounting only for amplitudes within third octave bands
- real-valued FFT - accounting only for the amplitude of each partial
- complex-valued FFT - accounting for both the amplitude and phase of each partial

2.2 Filter bank

In a live ensemble sound, there will be random beating on each partial. Given the well-known critical-band model of the auditory system, however, it may be that the resulting amplitude modulations need only be simulated per critical band. Hence we first tried a filter bank approach (also often called a band vocoder). This has the advantage of being conceptually straightforward, and easy to implement without problematic delays in the processing.

2.2.1 Analysis using filter bank

The analysis works by dividing an ensemble sound into a number of frequency bands and the amplitude within each band is recorded separately. (as in the analysis section of a band vocoder) "see figure 2.1 on page 5." The recorded amplitude envelopes can then be used for modulating a solo sound at a later time. Ideally the filter bands should resemble bark filters, with an asymmetric frequency response that is steeper toward low frequencies than toward high frequencies. As an approximation a bank of third octave filters, with the lowest bandpass filter modified to be a lowpass filter, can be used. A bark scale filter bank might perform slightly better, but third octave filter banks are more readily available. A potential problem with this approach is that since the filter bank has logarithmic spacing between the bands, at higher frequencies several harmonics will fall into one band. The measured amplitude within that band

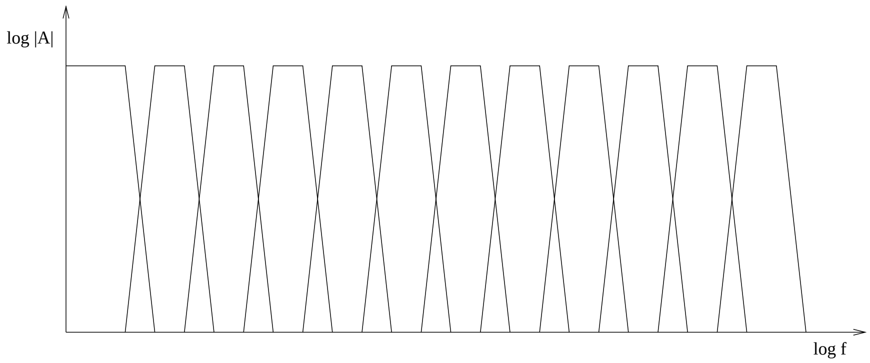


Figure 2.1: Typical filter bank

will then correspond to the amplitude of the sum of those harmonics. If for example two harmonics with 100% modulation are summed together, the sum will have much less than 100% modulation, provided that the modulations are uncorrelated. The amplitude detection method is important. It consists of a rectifier and a lowpass filter in series. If the filter cutoff frequency is set too low, only very slow modulations can be detected. If it is set too high, the signal will "leak" through.

Several things limit the information we can extract using this analysis; however, what remains may or may not be sufficient to produce perceptually adequate results. According to the critical band theory this should work, but problems may arise if the amplitude of the individual partials changes very rapidly.

2.2.2 Modulation using data from a filter bank

The modulation works in a similar way to the analysis. A solo sound is also divided into a number of frequency bands using a filter bank. The filter bank is exactly like the one used for analysis, but with added gain elements, one for each band, which then sum into one output. The gain of each band in the bank is modulated by the recorded amplitude envelope from the analysis. For this to work we must compensate for the fact that different source sounds have different characteristic spectrum envelopes. In applying the amplitude envelopes of individual frequency bands we must normalize the data, so that each band will have a gain of unity. If not, the resulting sound will be coloured by the spectrum envelope of the analysed ensemble sound. A potential problem is that if there is very little difference between the highest and lowest amplitude value within a specific amplitude envelope, then the quantisation noise will be significant when that envelope is normalized. The solo sound should have a pitch fairly close to that of the analyzed ensemble sound. If the pitches differ too much, the modulation will be more rapid or slower than intended, because the fundamental might move into another band.

2.2.3 Results using filter bank

The filter bank approach was tested in practice using an *Aladdin*[1] model. "see figure 2.2 on page 7." The filters used were derived from a twelve band third-octave telecom model, modified such that the lowest band was lowpass rather than bandpass. The cutoff frequency of the lowest filter was 500 Hz, and the cutoff frequency of the highest bandpass filter was 6400 Hz. The amplitude envelopes sampled from the twelve filters was stored to disk, and later normalized manually using a signal editor. The amplitude envelopes were then applied to a singing voice, first monaurally. The result does have an ensemble quality, but it is still quite obvious that one is listening to a single voice. Secondly, a stereo test with "independent" modulation in the left and right channels was conducted. The amplitude data was manually time shifted using a signal editor, and applied to the right channel, whilst the non shifted version was applied to the left channel. This improves the impression of ensemble, but the sound of a single voice still remains, especially when comparing the sound to a real choir. Perhaps this could be improved somewhat by adding more bands in the low frequency range, but our impression is that the lack of realism using this filter bank approach is not connected with the low frequencies. Rather there seemed to be a lack of small pitch deviations. This was confirmed visually. "see figure 2.3, 2.4 and 2.5" These limitations of the filter bank approach prompted us to explore more detailed techniques, based on FFT analysis and resynthesis.

2.3 Using the Fourier transform

The Fourier Transform works like a filter bank, in which the spacing between the bands is linear. The magnitude and phase within each band can be obtained from the real and imaginary parts produced by the algorithm. For studying the envelopes of the individual partials with a minimum of computation, it would be convenient to obtain a compact spectral representation with exactly one data point per partial. This would work like a filter bank with one band aligned to each partial. The most straightforward method is to use the Discrete Fourier Transform (DFT) with a window size of exactly two periods of the source signal. Since the DFT is very slow, we would like instead to use the Fast Fourier Transform (FFT). A property of the FFT is that it must have a window size of 2^n. Since we cannot align the FFT window size, we instead align the input data to the FFT. This can be done by resampling the input signal such that two periods fit exactly into one FFT window. This also minimizes the problem that arises if the length of a period of the source sound is not an integer number of samples. The DFT method will in that case have to round the length to an integer of samples, whilst the FFT method of resampling the data will distribute the error evenly across all periods. In practice an approximated resampling will be used, because of the amount computation otherwise needed. Both these methods assume that the source sound is harmonic, which is true in the case of solo string or voices.

The actual FFT / inverse FFT algoritm is beyond the scope of the report, but a tutorial can be found in Jaffe [6] and [7].

[1] *Aladdin Interactive DSP* by AB Nyvalla DSP

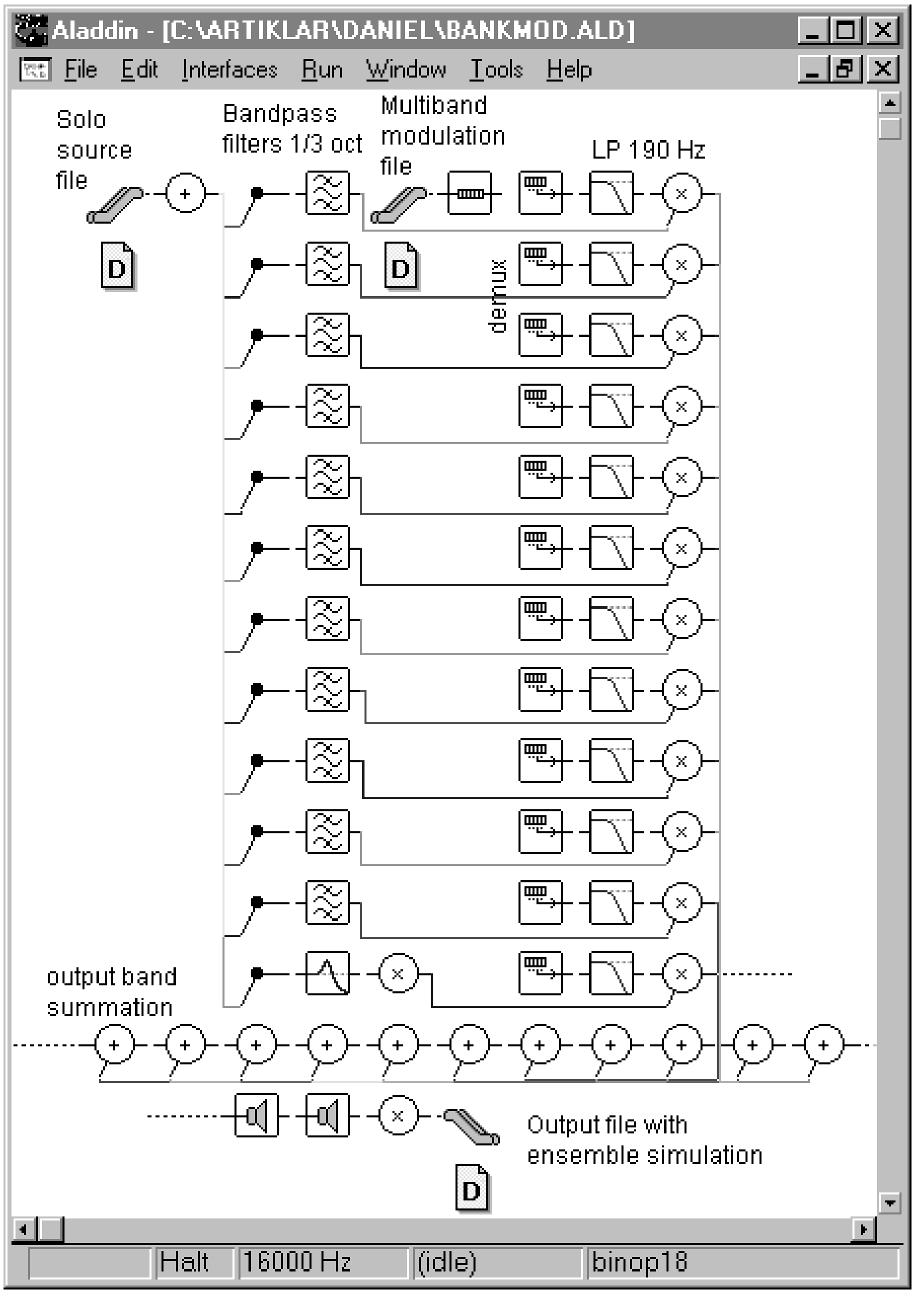


Figure 2.2: Aladdin filter bank model

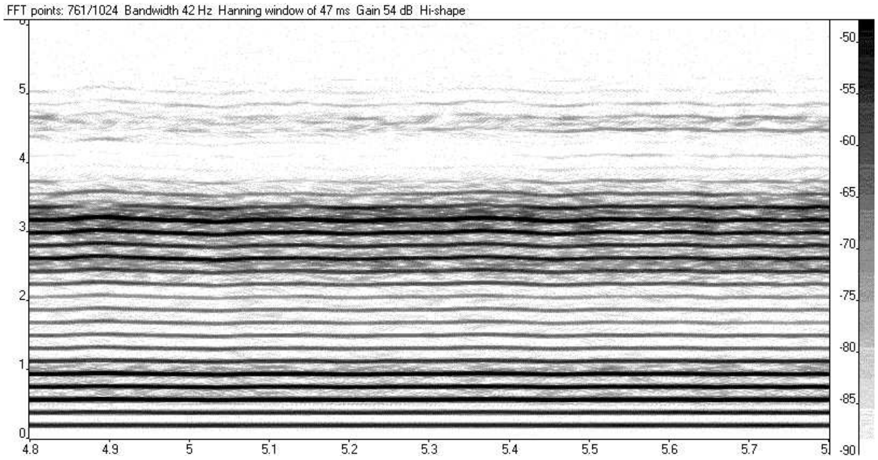


Figure 2.3: Spectrum of input voice

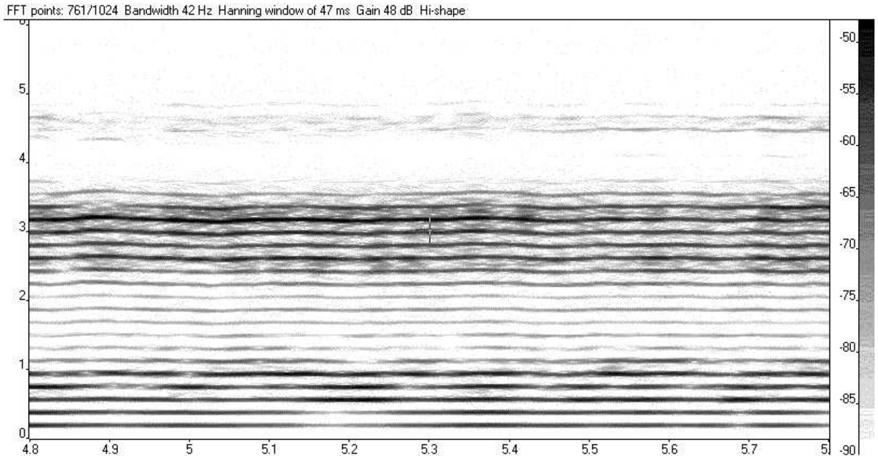


Figure 2.4: Spectrum of filter bank ensemble

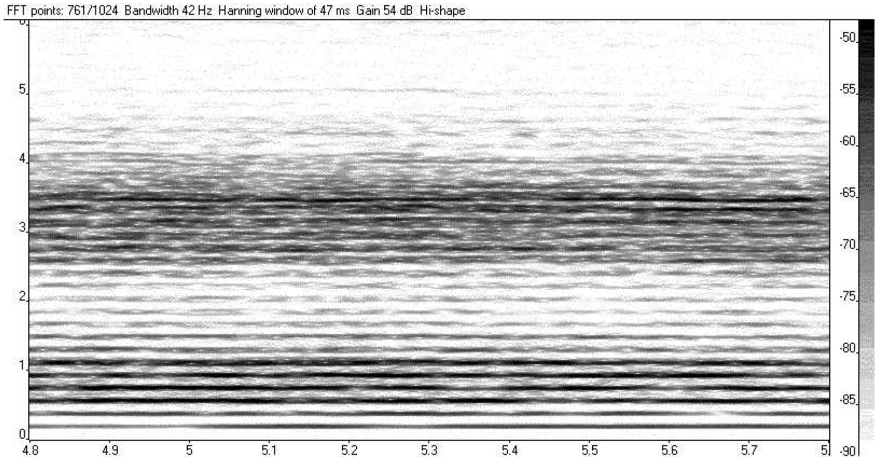


Figure 2.5: Spectrum of real ensemble

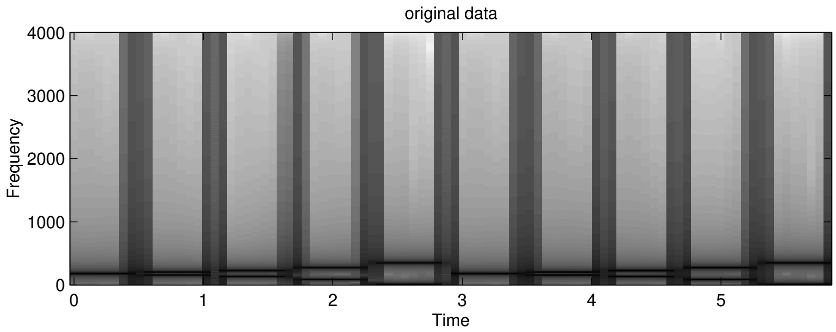
2.3.1 Theory for using FFT

The idea is to split the data into n data streams of bandwidth f_0, where n is the number of partials in the sound. The data is analyzed by first resampling it so that two periods of the fundamental fit perfectly into the FFT size (N). Then the data is Fourier transformed using the well known FFT algorithm. The spacing between the elements in the FFT vector should now be f_0, i.e., one harmonic for each element. The first element is the DC component, and the rest are: fundamental, partial 2, partial 3, and so on. The maximal bandwidth of the amplitude data may be equal to f_0 on both sides. In other words, the amplitude of a given partial must not change too quickly. In the frequency domain this would have the consequence that the sideband from one partial will overlap the next band. The sample rate of the amplitude envelope will have to be at least $2f_0$. This is no problem for the first few harmonics, but because of the increasing modulation it may become an issue for the higher harmonics. At what frequency might sideband leakage become a problem? This would depend on a number of factors including: frequency scattering in the original signal, the fundamental (f_0) and the windowing function employed for the Fourier Transform. A rough estimate shows that this may inflict problems around partial 50-100, perhaps even lower. This limitation must be kept in mind when evaluating the result of the analysis, possibly discarding the highest partials.

2.3.2 Keeping what is important

If we compute an FFT of a signal s_n and want to be able to resynthesize it we need to save the spectral data. If the number of points in the FFT are N and $s_{[0,N-1]}$ is real, then only the first $N/2+1$ complex points need to be kept.

$$
\begin{array}{l}
p_n = \mathrm{FFT}(s_n, N) \\
\text{if} \quad s_n \in \mathbf{R} \quad \text{for} \quad 0 \leqslant n < N \quad \text{then} \quad p_{N-n} = p_n^* \quad \text{for} \quad N/2 < n < N
\end{array}
$$



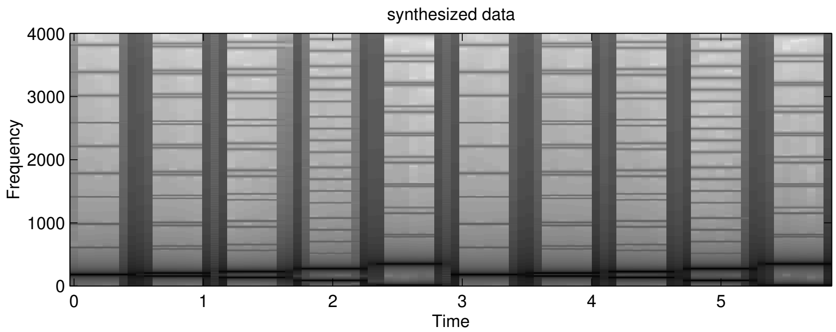


Figure 2.6: Test tones analyzed and resynthesized; the first 5 test tones are in phase, and the next 5 have different phase.

2.3.3 Overlapping FFT and IFFT

To obtain a time series of the amplitude values of each harmonic, we slide the FFT window along the input data in overlapping steps. As windowing function we use the square root of a hamming window. The overlapping must be $N/2$ because of the windowing.

When resynthesizing the data an inverse FFT (or IFFT) algorithm is used. After inverse transforming we window the data once more with the square root of a hamming window. The total window function is then equal to a regular hamming window, and if the data is put together using the same overlapping ($N/2$) as before, the result will be the original input data. "see figure 2.6 on page 10." Some experiments with a regular hamming window applied both during FFT and IFFT and the overlapping set to $N/4$, seem to provide a better result. The choice of windowing function and overlapping is non-trivial, and there may be better combinations than those above.

2.4 Real-valued FFT

2.4.1 Analysis using real-valued FFT

The real-valued FFT appoach is better than the filter bank in that it does not try to approximate the amplitude envelopes. Recall that with the filter bank, several partials may fall within one band. Analysis using a real-valued FFT is similar to the filter bank approach in that it uses only amplitude information.

The major difference is that the FFT works like a linear filter bank. Properly aligned, one amplitude value per harmonic of the input sound is obtained. "see section 2.3.1 on page 9." If this is applied using a sliding window, one amplitude envelope per harmonic is obtained. A drawback is that more data is generated, and the problem of detecting rapid amplitude variations (as with the filter bank) remains.

2.4.2 Resynthesis/modulation using data from real-valued FFT

To resynthesize the data we apply the envelope of each analyzed partial to a sinusoid of the corresponding frequency. The purpose of the resynthesis is to verify that we have not lost too much information in the process.

To modulate using the data we apply the amplitude envelopes of the analyzed signal to a different source solo signal.

2.4.3 Results using real-valued FFT

The real-valued FFT approach was tested using the *sndan*[2] package for Unix systems. As test data, binaural recordings on analog tape of a male choir singing unison vowels were used. These recordings were sampled in stereo at 32 kHz, and the longest sustained vowels (about six seconds) were selected for analysis. "see table A.1 on page 20." The program `pvan` analyzes data by resampling it and then computing FFT in overlapping windows. In order to make the harmonics fit the bands of the FFT, the average fundamental frequency f_0 had to be determined, which we did using the *sndan* tools `mqan` and `fcheck`. The drift in mean f_0 was assumed to be zero over the six seconds. Each partial's amplitude envelope was tracked using the program `pvan` (which also tracks the frequencies of the partials; not used here).
The result of the analysis process was an array of amplitude vectors, one for each partial. The vectors contained one amplitude value per overlapping FFT window. This data was then read into Matlab and used to modulate a harmonic series of sinusoids with an amplitude of one. This procedure amounts to resynthesis of the original data, but with all phase information discarded. Furthermore, the amplitude modulations obtained in the analysis are fairly slow, due to the rectify-and-smooth detection principle. A consequence of these two limitations is that small frequency variations in the original are lost, resulting in a sound with reasonable amplitude modulations but with a perfectly static pitch. This result was deemed unsatisfactory and prompted us to continue with the complex-valued FFT.

In the first trial using real-valued FFT, the amplitude variations of the first 20 partials were applied to sine waves with frequencies at integer multiples of the measured f_0. In principle this should give a reconstruction of the original sound with only the phase information missing. Unfortunately we were restricted to an 8kHz sampling rate because of memory considerations. When comparing the result to the original signal lowpassed at 4kHz, they seem similar, but clearly a larger bandwidth was needed.

[2] *sndan* by James Beauchamp

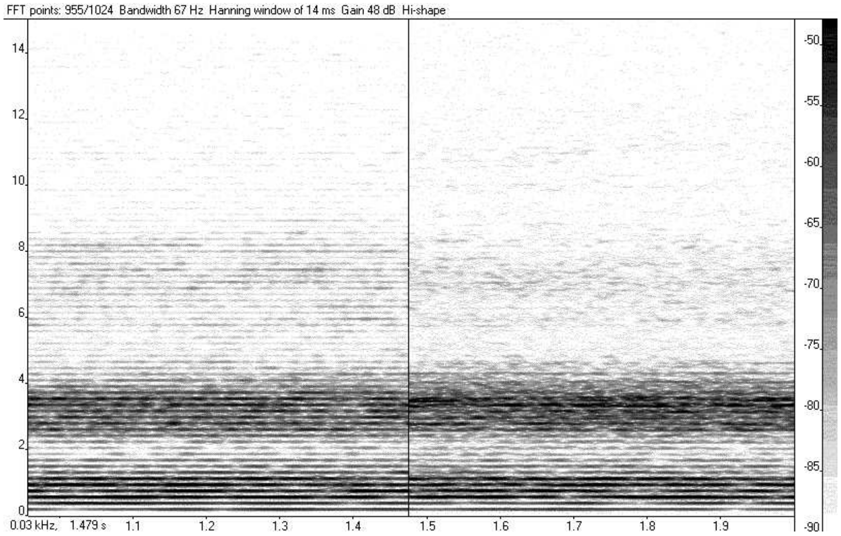


Figure 2.7: Spectrum of real-valued FFT resynthesized ensemble (left), and original ensemble (right).

For the second trial, more memory had been installed so that all partials within the range of the sample rate could be used. This made it apparent that the result has a static pitch, which is undesireable.

In the third trial, a simple test of cross applying data was made, by extracting the vowel spectrum of one unison choir and then applying it to a time-varying version. The motive for this test was to hear what happened when changing the relative long time average amplitudes of the individual partials. This might show if the partials were dependent of each other. The result is an acceptable ensemble, but it shows the same defects as the result of the previous trial.

An additional test was performed by applying the data extracted during analysis in stereo. Left channel data was correctly applied, while the right channel data was applied backwards. The idea is to create an "independent" modulation for each channel. This works fairly well to create a false stereo effect.

2.5 Complex-valued FFT

2.5.1 Analysis using complex-valued FFT

When analyzing using the real-valued FFT, we discarded the phase information of the signal. This is not a problem when the amplitude of the studied partial is constant or changes only slowly, but when the amplitude modulation of a partial becomes "fast", the difference between amplitude and frequency modulation becomes smaller. One consequence of disregarding the phase information is that the spreading of the signal around each FFT band centre always will be symmetric[3]. If we use the complex-valued amplitude instead, from which we can obtain both magnitude and phase, we can resynthesize the data with an asymmetric spreading. The exact reason for this is beyond the scope of this report. Apart from the preservation of complex-valued amplitude data, the complex-valued FFT method is no different from the real-valued FFT method.

2.5.2 Resynthesis/modulation using data from complex-valued FFT

The resynthesis and modulation using data from complex-valued FFT is similar to using data from a real-valued FFT, except in that it uses complex-valued modulation data instead of real.

2.5.3 Results using complex-valued FFT

The data is analyzed using the program `specan.m`. It takes a sound file, resamples it so that two periods will fit into one FFT of a convenient length. Then it records the FFT in overlapping windows. This is very similar to how `pvan` works, but here we also record the phase. We also need to know the fundamental frequency of each sound file. The spectral data is then resynthesized using `respect.m`. The selection of windowing function and overlapping is apparently non-trivial. The first attempts were made using a Hamming window, and an overlap of $N/4$, but that did not work well if the spectral data was modified too much. Later we switched to using the square-root of a Hamming window when FFT:ing, the same after IFFT:ing, and using an overlap of $N/2$. It is difficult to tell which selection was better.

The first trial consisted of analyzing several ensemble sound files, resynthesizing, and comparing the result to the original sound. This is similar to the first and second trial using the real-valued FFT method. The result is hardly distinguishable from the original. This shows us that the complex-valued FFT method is a nearly loss less method of converting a sound into the frequency domain, and back. However it does not tell if the frequency domain representation is correct. What happens if we modify the spectral data, and then resynthesize?

The second trial was an attempt to apply the time varying amplitude envelopes from a choir to that of a single singing voice. The time varying spectrum of `flack1.wav` was applied to `sundb1.wav`. They have approximatly the same f_0, which should minimize artifacts due to shifting of the spectrum. When we combined the average spectrum envelope of `sundb1.wav` with `flack1.wav` the result sounded alot like a choir of `sundb1.wav`, but when the amplitude envelopes were multiplied together, i.e accounting for the time variations in both sounds, the result was somewhat disappointing. It sounded like an ensemble, but with the character of a ring modulation effect superimposed. We assume this was due to problems with the windowing during resynthesis.

[3] Hint: think of the FFT as a heterodyne radio receiver, where the signal is mixed down with the band centre frequency and thereby is transposed down in frequency such that zero Hz becomes the new band centre.

2.5.4 Cross-synthesis

By cross-synthesis we mean taking the normalized (complex-valued) amplitude envelopes from an ensemble sound and applying them to a solo sound. The ensemble sound was analyzed using the Matlab program `specan.m`. This program resamples the signal and computes a complex-valued FFT, as described above. For each partial, the data was then normalized to the maximum magnitude to extract only the modulations, and discard the static spectrum envelope. This requires an ensemble spectrum that is as flat as possible, for example using the vowel [a:], or there will be too little information in the upper part of the spectrum.
A solo sound was then transformed to the frequency domain similarly to the ensemble sound, but without normalization. The normalized spectral data from the ensemble sound was then multiplied with the spectral data from the solo sound, in the complex domain. This procedure is often called "quadrature modulation". The effect of this operation is that the spectral spreading of each of the partials in the ensemble sound will be shifted to the instantaneous frequency of the corresponding partial in the solo sound.
The result of the quadrature modulation was then transformed back to the time domain, with the complementary Matlab program `respect.m`. A typical result gives an acceptable ensemble effect except for the tone attack and decay, which by design are outside the scope of this model. There is some "frizzle" at high frequencies; we believe this is due to quantization noise that becomes enlarged when normalizing the amplitudes of very weak partials in the ensemble sound. When the topmost octave was filtered out, this distortion was greatly reduced.

2.6 Modelling

The purpose of modelling is to design a function of the (complex-valued) amplitude variation of each partial that is similar to the natural variations in ensemble sounds. Ideally there should be a parameter controlling the degree of ensemble sensation to apply to the input solo sound.

2.6.1 Simple noise model

The first test on synthesis using a model is based on just amplitude modulating each partial with lowpass-filtered noise. The bandwidth was proportional to the frequency of each partial. This results in a sound that sounds typically noisy, and not at all like an ensemble. This suggests that lowpass-filtered noise may not be adequate as model. This also suggests that the amplitude variations might not be just uncorrelated noise as was assumed earlier.

2.6.2 Modelling the amplitude modulation functions

This part of the analysis consists of determining the probability function for each harmonic. Hopefully we should see a trend in this, such that we can make a probability model. One could guess that f_0 does not matter so much. However several problems are present. One is the fact that we have source sound samples of vowels, so the frequency response is coloured. We thus have to normalize the data, to rule out the effect of this. This operation raises the noise floor significantly (due to quantization), especially at the higher frequencies where the signals of musical sounds have little energy. The quantization noise floor makes it hard to "see" the actual data. Normalization will remove the colouration that the room introduces at the same time it removes the vowel spectrum. This will probably have to be approximated by hand afterwards for a realistic ensemble sound. The source of these problems lies in the small amount data we have available. Each 6 second sample produces very little data for each harmonic (typically 1500-3000 amplitude values). This will only allow us to compute one or two FFT windows, and that will make the output unreliable. More data is needed.

Chapter 3

Discussion

With reference to Dolson's result that amplitude modulations are sufficient for perception of ensemble, we now suggest that this assertion can be further qualified as follows: Pure amplitude modulations alone cannot simulate all aspects of the ensemble sound. The reason is that the notion of a signal amplitude is tied to a time window of measurement that is too long to capture the spectrum-smearing properties of an ensemble at high frequencies.

There remains to propose a model for the partial modulations in the complex-valued case. A typical 6 second recording of an ensemble only gives us about 1500-3000 amplitude envelope data points per partial. A stringent analysis would require a lot more data. This does not necessarily have to be the same sustained vowel, multiple independent recordings of a choir singing the same sustained vowel at the same pitch would do. A loop on the other hand, would not be sufficient.

The recordings of unison choirs were made using binaural microphones. In these experiments, the left and right channels were added to obtain a mono file prior to analysis. This in itself makes the sound a little bit different.

The resampling scheme which we employed incurs a lot of computation, but was chosen because we wished for a convenient representation of the modulation of each partial in an ensemble sound. Instead of resampling the input signal to align it to f_0, one might use a conventional FFT and try to find a model for how the spectral lines are blurred.
The inherent delay in computing the FFT might also be problematic, especially if a sound having a low f_0 is to be processed. The theory could be extended to splitting the FFT into parts. Allowing lower frequencies to have a longer processing delay than higher frequencies. This is left for others to try.

Chapter 4

Summary

The three methods described here have different advantages and drawbacks.

The filter bank is easy to understand and implement, it has neglible inherent delays in the processing, and does impart a certain ensemble sensation; but the quality is a bit synthetic.

The real-valued FFT method yields and uses information on each partial and therefore has greater potential for precision than the third-octave filter bank. However, f_0 must be determined separately.

The complex-valued FFT method is the general form of the real-valued FFT method. It gives complete control in the sense that the original signal can be reconstructed from the spectral representation. However, it is more difficult to describe the complex-valued amplitude envelopes with a model. The complex-valued FFT approach will probably need slightly more computation due to the complex-valued amplitude envelopes. The actual FFT needs less computation in the complex-valued case, as we do not need to calculate the magnitude of the FFT data points.

A full model using the complex-valued FFT method would have to include f_0 tracking and adaptive resampling. Nevertheless, this is the most promising of the three methods for a high-fidelty simulation. It is definitely worth investigating further.

- Filter bank
  - \+ Easy to understand and implement.
  - \+ Gives a sense of ensemble.
  - \+ neglible inherent processing delay
  - − The original solo voice remains easy to identify.
- Real-valued FFT
  - \+ Better sense of ensemble than the filter bank.
  - − f_0 must be known.
  - − inherent processing delay (one FFT window must be known)
- Complex-valued FFT
  - \+ Gives the best sense of ensemble.

  + nearly lossless spectral representation
  − Hard to implement.
  − f_0 must be known.
  − inherent processing delay (one FFT window must be known)

- Modeling the data
  + Does not need sampled modulations from a choir.
  − It is difficult to make a good model. (not enough reference data)

Appendix A

Sound data

A.1 Ensemble material

The ensemble material was recorded by Sten Ternström on 7.11.1984. It consists of Teknologkören's male section singing sustained Swedish vowels at different pitches. On parts of the tape the female section of the choir can be heard rehearsing in a nearby room. The room used is a normal lecture room at KTH[1], and has moderate reverbation.
Equipment used for recording:

- Sennheiser MKE2002 artificial head placed mid choir.
- Revox B77 tape machine (19cm/s)

The material was sampled from reel tape by Daniel Kahlin on 20.5.1998 using the following equipment:

- Revox A77 tape machine (19cm/s)
- Loughbourough Sound Images LSI-C32 sound card (32kHz stereo)

The sampled data was then split[2] into sound files according to table A.1 on page 20.

A.2 Solo material

The solo material was recorded by Sten Ternström (date unknown). It consists of two male singers and one female singer, one at a time, singing sustained Swedish vowels at different pitches. The material was transferred digitally (32kHz mono) by Daniel Kahlin from DAT, fall 1998. The sampled data was then split into sound files according to table A.2 on page 20.

[1] The Royal Institute of Technology, Stockholm, Sweden
[2] The ambient noise was left on purpose, so you know how much there is.

Name	Length	f_0	Comment
for1.wav	1430572	96 Hz	for
faar1.wav	1554476	126 Hz	får
faar2.wav	1342508	126 Hz	får
faang1.wav	1316908	157 Hz	fång
far1.wav	1465388	186 Hz	far
foer1.wav	1317932	95 Hz	för (click on mastertape)
fyr1.wav	1488940	128 Hz	fyr
forigen1.wav	1379372	159 Hz	for
fyrigen1.wav	1325100	191 Hz	fyr
fyrigen2.wav	1412140	191 Hz	fyr
fat1.wav	1498156	114 Hz	fat
flack1.wav	1298476	150 Hz	flack
fartva1.wav	1141804	189 Hz	far
fartva2.wav	1347628	189 Hz	far
fann1.wav	1397804	226 Hz	fann
ful1.wav	1268780	114 Hz	ful
fel1.wav	1177644	154 Hz	fel
fett1.wav	1321004	189 Hz	fett
fen1.wav	1256492	228 Hz	fen
omfor1.wav	1238060	97 Hz	for
omforig1.wav	1549356	164 Hz	for
omflack1.wav	1161260	153 Hz	flack
omfel1.wav	1269804	155 Hz	fel
info.wav	1640492		Sten Ternström speaking

Table A.1: These are the original ensemble sound files.

Name	Length	f_0	Comment
sundb1.wav	404524	150.8 Hz	Johan Sundberg “i”
sundb2.wav	264236	149.8 Hz	Johan Sundberg “o”
sundb3.wav	391212	146.2 Hz	Johan Sundberg “a”
sundb4.wav	661548	147.4 Hz	Johan Sundberg “e”
sundb5.wav	563244	152.1 Hz	Johan Sundberg “i”
terns1.wav	629804	216.8 Hz	Sten Ternström “å”
terns2.wav	657452	218.4 Hz	Sten Ternström “i”
terns3.wav	615468	216.6 Hz	Sten Ternström “o”
terns4.wav	676908	216.1 Hz	Sten Ternström “e”
rydin1.wav	541740	608 Hz	Cecilia Rydinger “å”
rydin2.wav	440364	609 Hz	Cecilia Rydinger “a”

Table A.2: These are the solo sound files.

Appendix B

Matlab source code

B.1 pvtest.m

```
function pvtest(name1, f1, name2, f2, nameout, fout);
% PVTEST - cross apply spectrums
%
% usage: pvtest(name1, f1, name2, f2, nameout, fout);
%   name1    - name of ensemble sound (wav file)
%   f1       - fundamental of ensemble sound
%   name2    - name of solo sound (wav file)
%   f2       - fundamental of solo sound
%   nameout  - name of output sound
%   fout     - fundamental of output sound
%
% $Id: pvtest.m,v 1.13 1999/03/01 16:56:05 daniel Exp $
% Copyright (c) 1998, 1999 Daniel Kahlin <daniel@kahlin.net>

% if this is we apply the _mean_ spectrum of 'name2'
% to 'name1'
medel=0;


% analyze data
data = specan(name1,f1);

% analyze data
data2 = specan(name2,f2);

% normalize data
for I=1:size(data,1),
  nm=max(abs(data(I,:)));
  data(I,:) = ( data(I,:) / nm ) ;
  data(I,1)=0;
end
```

```
if medel
  % calculate the spectrum
  spec2 = sum(abs(data2),2);

  % apply spectrum to datastream
  dataut=zeros(size(data));
  for I=1:length(data),
    dataut(:,I) = data(:,I) .* spec2;
  end
else
  % apply spectrum to datastream
  utlen=min([length(data) length(data2)]);
  dataut=zeros([size(data,1) utlen]);
  for I=1:utlen,
    dataut(:,I) = data(:,I) .* data2(:,I);
  end
end

% resynth data
respect(dataut,fout,nameout)


%
% EOF
%
```

B.2 specan.m

```
function data = specan(name,f0,Fs);
% SPECtral ANalysis of sample
%
% usage: data = specan(name,f0,Fs);
%   name     - name of wav file to be analyzed (May be a vector)
%   f0       - the fundamental of the sound
%   Fs       - optional samplerate
%
% $Id: specan.m,v 1.6 1999/03/01 16:56:06 daniel Exp $
% Copyright (c) 1998, 1999 Daniel Kahlin <daniel@kahlin.net>
if (nargin<2)
     fprintf('specan: too few arguments\n');
     return;
end

if (nargin<3)
     Fs=32000;
end

% set parameters
inname=name;
```

```
freq0=f0;

% This is the FFTLEN to contain 2 periods of freq0
FFTLEN=512;

% set this if you want exact resampling (should always be!)
% If we have a weird f0 this might take a while
exact=0;

% set this if you want the output normalized to |Y|=1
normalize=0;

%
% Read sound
%
if inname
  [Y,Fs,bits]=wavread(inname);
  [nsmpls nchnls]=size(Y);
  fprintf('loaded %s  Fs=%d Hz  res=%d bits  channels=%d\n', inn
ame, Fs, bits, nchnls);
  Y=monofy(Y);
else
  Y=monofy(name);
  bits=16;
end

%
% resample to perfectly fit into fft
%
fprintf('resampling...');
if exact
  % cutdown to a perfect multiple of samplerate
  per=Fs/freq0;
  numb=floor(length(Y)/per);
  nbefore=floor(numb*per);
  Y=Y(1:nbefore);
  nafter=FFTLEN*numb;
  Y=resample(Y,nafter,nbefore);
else
  % approximate sample ratio
  oldfftlen=round(Fs/freq0);
  nbefore=length(Y);
  Y=resample(Y,FFTLEN,oldfftlen);
  nafter=length(Y);
end

Fsafter=(nafter/nbefore)*Fs;
nshould=round(FFTLEN*freq0*nbefore/Fs);
fprintf('\rresampled %d samples into %d (exact=%d)\n',nbefore,na
fter,nshould);
```

```
%
% Process
%
FRACTION=4;
STEP=FFTLEN/FRACTION;
HLEN=FFTLEN/2;
FRAMES=floor((nafter-FFTLEN)/STEP);
data=zeros(HLEN+1,FRAMES);

fprintf('computing %d frames of data...',FRAMES);
POS=1;
for I = 1:FRAMES,
  buffer = Y(POS:(POS+FFTLEN-1)) .* sqrt( hamming (FFTLEN) );
  fftout=fft( buffer, FFTLEN );
  data(:,I)=fftout(1:HLEN+1);
  POS=POS+STEP;
end
fprintf('Ok.\n');


%
% scale data
%
if normalize
  for I=1:HLEN+1,
    nm=max(abs(data(I,:)));
    data(I,:) = ( data(I,:) / nm ) ;
  end
end

%
% EOF
%
```

B.3 respect.m

```
function Y=respect(data,f0,name,Fs);
% REsynthesis of SPECTral data
%
% usage: respect(data,f0,name,Fs);
%   data     - the spectral data
%   f0       - the fundamental of the sound
%   name     - name of output wav file (if omitted the array wil
l be returned)
%   Fs       - optional samplerate
%
% $Id: respect.m,v 1.8 1999/03/01 16:56:05 daniel Exp $
```

```
% Copyright (c) 1998, 1999 Daniel Kahlin <daniel@kahlin.net>
if (nargin<2)
     fprintf('respect: too few arguments\n');
     return;
end
if (nargin<3)
  name=[];
end
if (nargin<4)
  Fs=32000;
end

% set parameters
outname=name;
freq0=f0;
bits=16;

% This is the FFTLEN to contain 2 periods of freq0
FFTLEN=(size(data,1)-1)*2;

% resample to perfectly fit into fft
oldfftlen=round(Fs/freq0);

FRACTION=4;
STEP=FFTLEN/FRACTION;
HLEN=FFTLEN/2;
FRAMES=length(data);


%
% Resynthetisize to verify data
%
fprintf('resynthetisizing %d frames of data...',FRAMES);
Ynew=zeros(FRAMES*STEP+FFTLEN,1);
POS=1;
for I = 1:FRAMES,
  fftin=[data(1:HLEN+1,I); conj(flipud(data(2:HLEN,I)))];
  buffer=real(ifft(fftin, FFTLEN)) .* sqrt ( hamming (FFTLEN) );
  Ynew(POS:POS+FFTLEN-1)=Ynew(POS:POS+FFTLEN-1)+buffer;
  POS=POS+STEP;
end
fprintf('Ok.\n');


Ynew=resample(Ynew,oldfftlen,FFTLEN);

% normalize
Ynew = rescale(Ynew,0.9);

if outname
```

```
  % Write sound
  wavwrite(Ynew,Fs,bits,outname)
  fprintf('wrote %s  len=%d samples  Fs=%d Hz  res=%d bits\n', o
utname, length(Ynew), Fs, bits);
else
  % return value
  Y=Ynew;
end

%
% EOF
%
```

Appendix C

Acknowledgements

Thank you to Sten Ternstöm who was my supervisor on this project, and was very helpful in general.

We are grateful to Johan Liljencrants for in-depth discussions.

Thank you to Theresa Sandin who, with love, motivated me to finish this report.

Bibliography

[1] Mark Dolson: *A Tracking Phase Vocoder and Its Use in the Analysis of Ensemble Sounds*, pp 83-121, Ph. D. Thesis, Caltech, 1983

[2] Takeshi Adachi: *US Patent 3,866,505, Ensemble effect imparting device using a bucket brigade device for an electric musical instrument*, Hamamatsu, Japan, 1975

[3] Cotton, Jr. et al.: *US Patent 4,384,505, Chorus Generator System*, 1983

[4] Sten Ternström: *Perceptual Evaluations of Voice Scatter in Unison Choir Sounds*, Journal of Voice Vol. 7 No. 2. pp. 129-135, 1993

[5] Sten Ternström and Anders Friberg: *Analysis and simulation of small variations in the fundamental frequency of sustained vowels*, KTH Speech Transmission Laboratory Quarterly Progress and Status Report `STL-QPSR 3-1989`, pp. 1-14, 1989

[6] David A. Jaffe: *Spectrum Analysis Tutorial, Part 1: The Discrete Fourier Transform*, Computer Music Journal, Vol. 11, No. 2, Summer 1987

[7] David A. Jaffe: *Spectrum Analysis Tutorial, Part 2: The Discrete Fourier Transform*, Computer Music Journal, Vol. 11, No. 3, Fall 1987

[8] Daniel Kahlin and Sten Ternström: *The Chorus Effect Revisited: Experiments in Frequency-Domain Analysis and Simulation of Ensemble Sounds*, Proc of Euromicro'99, IEEE Computer Society, `PR00321`, `ISBN 0-7695-0321-7`; 2: pp. 75-80, 1999