Computação Musical – 3º Exercício Programa
Prof. Marcelo Queiroz – Data de entrega: **5/7/2011**

Instruções: Este EP deve ser feito em DUPLAS. A ideia é que os dois alunos participem da construção de todo o EP, mas um aluno poderá se dedicar mais especificamente à parte A e o outro aluno à parte S. Estas duas partes estão fortemente relacionadas, embora seu desenvolvimento possa ser feito de forma independente.

Phase Vocoder

1 Introdução

Neste EP iremos construir em Puredata um mecanismo conhecido como *phase vocoder (PV)*, que permite codificar um arquivo de áudio como uma soma de osciladores senoidais com amplitude e frequência variáveis e, através desta descrição, ressintetizar o sinal original com modificações arbitrárias de andamento (velocidade de reprodução) e de altura musical (transposição).

Especificamente, o sinal de áudio $x(t)$ da entrada será re-escrito, de forma aproximada, como

$$x(t) \approx \sum_{k=1}^{K} a_k(t) * osc(f_k(t)),$$

onde as funções $a_1(t), \ldots, a_N(t)$ e $f_1(t), \ldots, f_K(t)$ serão obtidas pela análise de $x(t)$ em instantes amostrados $t = 0, \Delta_t, 2\Delta_t, \ldots, T\Delta_t$. Após a obtenção desta codificação do sinal, sua ressíntese será feita pela expressão

$$y(t) = \sum_{k=1}^{K} a_k(\alpha t) * osc(\beta f_k(\alpha t)),$$

onde α é um fator de escala temporal ($\alpha = 1$ corresponde à velocidade de reprodução original) e β é um fator de transposição da altura musical ($\beta = 1$ é a altura musical original).

As etapas de análise (A) e síntese (S) não serão síncronas (para permitir reconstruções mais rápidas do que o sinal original, com $\alpha > 1$), e usarão ferramentas matemáticas e computacionais diferentes. Estas duas etapas serão discutidas em detalhe nas seções a seguir, inicialmente através de uma exposição teórica, e em seguida através dos detalhes relativos à implementação em Puredata.

2 Análise usando FFT

A transformada rápida de Fourier é um algoritmo que permite a representação de *janelas* de N amostras do sinal como uma soma de senóides harmonicamente relacionadas. Vamos discutir a FFT aqui sob uma perspectiva instrumental, ou seja, em relação ao que é necessário saber para utilizar a saída de uma FFT pronta, como aquela fornecida pelo objeto `rfft~` do Pd.

Uma janela $(x_0, x_1, \ldots, x_{N-1})$ pode ser interpretada como um período de um sinal periódico; de acordo com esta interpretação, este fragmento de duração $\frac{N}{R}$ seg (aqui R é a taxa de amostragem) pode ser representado de maneira perfeita por uma família de senóides com frequências $0, f_0, 2f_0, \ldots$, onde $f_0 = \frac{R}{N}$ Hz corresponde a um período fundamental igual à duração da janela. Pela discretização do sinal, qualquer frequência da forma kf_0 acima da frequência de Nyquist $\left(\frac{R}{2}\right)$ sofreria rebatimento e seria idêntica

a uma outra frequência menor, de onde vemos que basta a coleção finita de senóides com frequências $0, f_0, 2f_0, \ldots, \frac{N}{2}f_0$ para representar a janela original.

A FFT desta janela de N pontos pode ser usada para obter as amplitudes e as fases iniciais destas senóides. Cada valor produzido pela FFT é um número complexo cuja magnitude representa a amplitude de uma senóide, e cuja fase representa a fase inicial desta senóide na janela em questão. Por construção, a FFT representa cada senóide real como a soma de duas senóides complexas, cada uma delas recebendo $\frac{N}{2}$ vezes a amplitude da senóide original. Isso é verdade para todas as frequências entre f_0 e $\left(\frac{N}{2} - 1\right) f_0$; as frequências 0 Hz e $\frac{N}{2}f_0 = \frac{R}{2}$ Hz são exceções, mas elas não serão usadas no PV.

Nosso objetivo no PV é obter uma representação de *amplitudes* e *frequências* variáveis para uma coleção de osciladores, porém o resultado da FFT são amplitudes e fases correspondentes *às frequências fixas* $0, f_0, 2f_0, \ldots, \frac{N}{2}f_0$, cujos períodos estão associados ao tamanho da janela de análise. Para poder representar o sinal da forma que queremos, é preciso traduzir de formas diferentes as amplitudes e as frequências resultantes do cálculo da FFT de cada janela. Em relação às amplitudes, podemos adotar o princípio de que, para cada janela, a amplitude do k-ésimo oscilador na representação do phase vocoder corresponde a $\frac{2}{N}$ vezes a magnitude do k-ésimo valor da FFT. Em relação às fases e frequências, a tradução se dá pela forma como a FFT representa frequências *diferentes* daquelas usadas na análise.

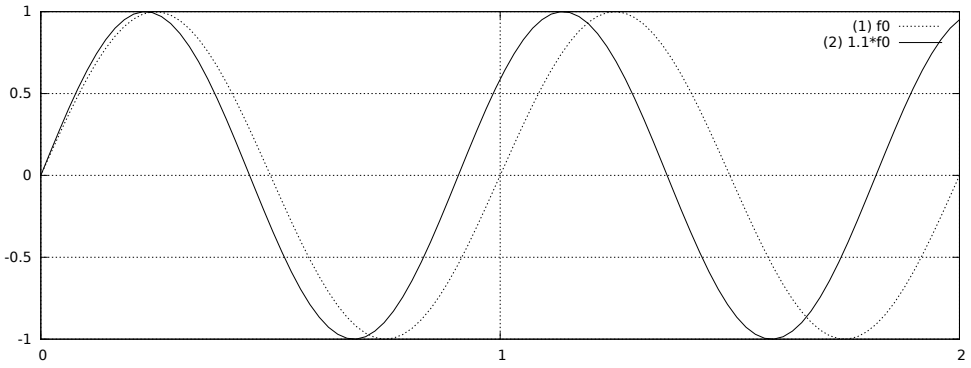

Figura 1: Representação de duas janelas consecutivas e dois sinais senoidais com frequências distintas: (1) f_0, a frequência fundamental da análise da FFT; e (2) $1.1f_0$, que será representado pelo primeiro oscilador do PV como uma variação em relação à frequência de oscilação f_0.

Para entender a relação entre a variação de fase inicial de um certo sinal e a possibilidade de representação deste sinal como uma variação de uma frequência de análise, considere o sinal senoidal da figura 1, com frequência $1.1f_0$, próxima da frequência de análise f_0. Na primeira janela, a fase inicial é 0, enquanto que na segunda janela a fase inicial é $\frac{2\pi}{10}$, pois o sinal já percorreu 10% da sua segunda volta pelo ciclo trigonométrico na janela anterior. Este acúmulo de fase caracteriza a diferença entre a frequência real do sinal ($1.1f_0$) e a frequência de análise usada pela FFT (f_0).

Desta forma, é possível utilizar a diferença de fase inicial de uma mesma frequência de análise em duas janelas consecutivas para estimar a frequência de uma componente real do sinal localizada naquelas janelas. Podemos estimar a frequência real do k-ésimo oscilador a partir da variação de fase inicial $\Delta_\phi(k)$ entre duas janelas sucessivas. Considerando que a distância entre os inícios de duas janelas sucessivas é de $\Delta_t = \frac{N}{MR}$ seg (com $M \geq 1$ para permitir sobreposição de janelas), a frequência real do k-ésimo oscilador será

$$f(k) = \left(\frac{M\Delta_\phi(k)}{2\pi} + k \right) f_0.$$

Em relação ao exemplo da figura, temos $k = 1$ (associado à frequência da janela), $M = 1$ (pois as janelas não têm sobreposição) e $\Delta_\phi(k) = \frac{2\pi}{10}$, de onde $f(k) = 1.1f_0$. Esta correção da frequência do k-ésimo oscilador em relação à frequência de análise kf_0 sempre produzirá um valor entre $\left(k - \frac{1}{2}\right) f_0$ e $\left(k + \frac{1}{2}\right) f_0$, desde que a variação $M\Delta_\phi(k)$ seja computada de forma a recair no intervalo $[-\pi, +\pi]$ (somando ou subtraindo 2π, se necessário).

Podemos, portanto, produzir amplitudes e frequências para uma coleção de $K = \frac{N}{2} - 1$ osciladores, uma vez para cada janela de análise do sinal da entrada, ou seja, a cada $\Delta_t = \frac{N}{MR}$ seg. Na seção 4 serão fornecidas dicas sobre a implementação deste processo em um *patch* Pd.

3 Síntese usando osciladores por consulta a tabela

Nesta seção veremos como podemos usar um conjunto de osciladores que lêem todos uma mesma tabela SENO, contendo uma discretização de um período completo da função *sen(x)* usando S amostras, para computar um sinal de áudio, amostrado a R Hz, que corresponde à expressão

$$y(t) = \sum_{k=1}^{K} a_k(\alpha t) * osc(\beta f_k(\alpha t)),$$

onde os parâmetros a_k e f_k são fornecidos em intervalos de tempo regulares $t = 0, \Delta_t, 2\Delta_t, \ldots, T\Delta_t$.

Considere o primeiro oscilador, definido pela expressão $a_1(\alpha t) * osc(\beta f_1(\alpha t))$. Para simplificar a discussão, vamos considerar a produção de $L = \frac{N}{\alpha M}$ amostras (arredondando L se necessário), correspondentes ao intervalo de tempo $t \in \left[\frac{j\Delta_t}{\alpha}, \frac{(j+1)\Delta_t}{\alpha}\right]$, e indexando estas amostras de 0 a $L - 1$.

Para que este oscilador não produza descontinuidades audíveis, todos os valores de amplitude devem ser interpolados; especificamente, vamos definir

$$a_1(n) = (1 - \lambda(n))a_1(j\Delta_t) + \lambda(n)a_1((j + 1)\Delta_t),$$

onde $\lambda(n) = \frac{n}{L}$. A frequência será mantida constante neste intervalo, ou seja, $\beta f_1(n) = \beta f_1(j\Delta_t) = \beta f_1$, e ela define a velocidade de varredura da tabela para este oscilador neste intervalo temporal, através da expressão

$$\Delta_1 = \beta f_1 \frac{S}{R}.$$

O valor do oscilador na amostra n será portanto SENO$[I_1 + n\Delta_1 \bmod S]$, onde I_1 é o índice inicial de leitura deste oscilador, e mod S representa o acesso circular do vetor. Todas as leituras serão feitas usando interpolação cúbica (de 4 pontos); especificamente, para acessar o vetor SENO numa posição fracionária $x \in [0, S)$ define-se os 4 pontos adjacentes $x_0, x_1 = \lfloor x \rfloor, x_2, x_3$ (todos mod S) e usa-se a expressão

$$-\frac{y(y-1)(y-2)}{6}\text{SENO}[x_0] + \frac{(y+1)(y-1)(y-2)}{2}\text{SENO}[x_1]$$
$$-\frac{(y+1)y(y-2)}{2}\text{SENO}[x_2] + \frac{(y+1)y(y-1)}{6}\text{SENO}[x_3]$$

onde $y = x - \lfloor x \rfloor$.

Observe que o conjunto de índices iniciais $I_1, \ldots, I_K$ também é importante para garantir a continuidade dos K osciladores nas transições entre segmentos temporais. No início do processo, podemos definir $I_j = 0$, $\forall j$. Ao final do processamento de um intervalo temporal, temos que definir os novos índices iniciais do próximo intervalo como

$$I'_j = I_j + L\Delta_j \bmod S.$$

A seção 5 detalhará a construção de um *external* para Pd para realizar a ressíntese do phase vocoder.

4 Especificação da Parte A

A etapa de análise e construção da representação do sinal através dos parâmetros do PV será feita obrigatoriamente combinando objetos prontos do Pd, além de um objeto auxiliar `fprintf~` para escrever valores em arquivos texto. Especificamente, construiremos um *patch* Pd com um único *inlet* capaz de processar as seguintes mensagens:

- `window N`: define o tamanho (potência de 2) da janela de análise, em amostras; considere como valor default $N = 2048$, assim esta mensagem não será de uso obrigatório;
- `overlap M`: define o fator (potência de 2) de sobreposição de janelas; considere como valor default $M = 2$, assim esta mensagem não será de uso obrigatório;
- `input arquivo.wav`: define o arquivo a ser analisado (parâmetro obrigatório);
- `output arquivo.txt`: define onde escrever o resultado da análise (parâmetro obrigatório); este arquivo conterá os valores N e M na primeira linha, e depois uma sequência de linhas com $K = \frac{N}{2} - 1$ valores cada, K valores de amplitude nas linhas pares e K valores de frequência nas linhas ímpares;
- `start`: inicia o processo de análise.

Seu patch deverá:

- cuidar da inicialização das estruturas de dados necessárias para a análise;
- processar as mensagens de inicialização, recebendo os parâmetros N e M que definem a análise e os nomes dos arquivos de entrada e saída; definir no Pd a blocagem do sinal em janelas de N amostras com sobreposição M; abrir o arquivo de saída e escrever os parâmetros N e M;
- iniciar o processamento de áudio e alimentá-lo com o conteúdo do arquivo de entrada; analisar o sinal através da FFT, computando os valores de amplitude e frequência para cada janela, e escrevendo-os na saída; desligar o processamento de áudio e gravar o resultado da análise em um arquivo texto.

O cálculo das magnitudes dos valores produzidos pelo objeto `rfft~` é feito pela expressão $\sqrt{a^2 + b^2}$, onde a e b são as partes real e imaginária da FFT. Use o objeto `atan2~` para extrair as fases. O cálculo das frequências dependerá da comparação da janela atual com a análise da janela anterior; não se esqueça que as diferenças de fase precisam ser trazidas para o intervalo $[-\frac{\pi}{M}, +\frac{\pi}{M}]$. Você precisará definir o tamanho do bloco de análise e o fator de sobreposição de janelas no seu patch, o que pode ser feito com o objeto `block~`. O controle do processamento de áudio pode ser feito tanto com o objeto `switch~` quanto com as mensagens `pd dsp 1` e `pd dsp 0`.

Para escrever valores produzidos pela análise em um arquivo texto, use o objeto `fprintf~`, que foi implementado como um external. Este objeto receberá dois sinais, um contendo as amplitudes e o outro as frequências, e escreverá em um arquivo texto uma linha com K amplitudes e outra linha com K frequências, a cada janela de análise. O arquivo `fprintf~help.pd` deve ajudá-lo a entender o uso deste objeto; o código-fonte também está disponível, para quem tiver interesse em saber como ele foi feito.

Para ajudá-lo nesta parte, consulte o patch I07 da documentação do Pd; este patch implementa um phase vocoder ligeiramente diferente do nosso, pois a ressíntese usa FFT e síntese granular, mas a parte de análise (subpatch fft-analysis) é similar (embora a nossa seja bem mais simples).

5 Especificação da Parte S

A etapa de ressíntese do phase vocoder será feita obrigatoriamente por um objeto criado através de um *external* em C para Pd. Este external deverá receber, através do primeiro *inlet*, o nome do arquivo texto com o resultado da análise (pela mensagem `open arquivo.txt`) e um disparo (*bang*) para dar início à ressíntese propriamente dita. Através do segundo e terceiro *inlets* o objeto receberá os parâmetros α e β correspondentes às modificações de tempo e frequência pretendidas. Uma característica importante deste external é o fato de que ele deve aceitar modificações dos parâmetros α e β também após o início do processamento; concretamente, devemos oferecer ao usuário a possibilidade de controlar estes dois parâmetros em tempo-real, por exemplo conectando *sliders* aos inlets correspondentes.

Para simplificar a implementação, vamos sincronizar a atualização dos parâmetros α e β com o início dos segmentos temporais correspondentes aos valores lidos do arquivo de entrada. Especificamente, ao receber uma atualização de α ou β, o inlet correspondente deverá armazenar este valor numa variável auxiliar; ao terminar o processamento de um segmento temporal, o código se encarregará de transferir os valores pendentes de α e β para as variáveis que serão usadas de fato na geração do sinal no próximo segmento.

Em relação à construção de um external em Pd, o código é escrito em C, usando definições do arquivo `m_pd.h`; a estrutura do external é inspirada no paradigma POO, e requer a definição das seguintes funções:

setup inicialização da classe, onde são definidos parâmetros globais de todos os objetos daquele tipo (como o nome das funções chamadas para processar cada inlet);

new inicialização do objeto, incluindo a criação de inlets e outlets, e definição de valores iniciais para os atributos (variáveis locais) do objeto;

bang/float/symbol/anything funções que processam cada inlet, de acordo com o tipo do valor recebido;

dsp coloca a função de processamento de áudio na lista do escalonador do Pd; e

perform chamada a cada ciclo de processamento de áudio do Pd para processar os sinais de áudio de entrada e saída.

Você pode construir o seu external a partir do template de exemplo chamado `fileosc~`. Este objeto lê um arquivo com uma lista de frequências e durações (em número de amostras), e produz a saída de um oscilador senoidal por consulta a tabela que segue uma pseudo-partitura definida por um arquivo texto de entrada, contendo pares de valores de `frequência` (em Hz) e `duração` (em amostras). Boa parte do que você deve fazer na parte S já está resolvido neste patch. Fica faltando considerar a soma de K osciladores, o tratamento das amplitudes e o tratamento dos parâmetros α e β. Você encontrará mais informações sobre externals no tutorial disponível neste endereço.

Observação importante: Todos os trabalhos devem ser entregues pelo PACA. Cada dupla deverá entregar um arquivo .tar ou .zip contendo o patch Pd correpondente à parte A e o código em C correspondente à parte S. Não entregue arquivos compilados ou exemplos de áudio ou análise.

Bom Trabalho!