Um mixer automático de música eletrônica

Ábner de Marcos Neves
DCC, UFMG
Belo Horizonte, Brasil
abnerneves@ufmg.br

Flavio Vinicius Diniz de Figueiredo
DCC, UFMG
Belo Horizonte, Brasil
flaviovdf@dcc.ufmg.br

***Abstract*—This work proposes the construction of an automatic electronic music mixer designed to execute the fundamental technical steps of a DJ set. While not intended to replace the artistic aspect of track selection, the system aims to provide a high-level tool for set preparation and to enhance the personal playlist listening experience. The methodology relies on a pipeline comprising rhythmic analysis, segmentation, time stretching, and beatmatching. To identify candidates for entry and exit points (hot cues), the algorithm utilizes novelty-based segmentation derived from Self-Similarity Matrices (SSM) and MFCCs. For tempo synchronization, the study compares distinct time-stretching libraries – librosa, rubberband, and SoX – ultimately selecting SoX for its superior spectral preservation. The system implements a dynamic beatmatching to address phase drift and executes three types of transitions: linear crossfade, logarithmic crossfade and bass swap. Results indicate that while the system produces smooth, synchronized transitions when rhythmic analysis is precise, the overall quality remains heavily dependent on the accuracy of the initial beatgrid detection.**

Keywords—automatic music mixing, beatmatching, music information retrieval, electronic music.

I. Introdução

DJ ou playlist? Do vinil ao pendrive, as facilidades tecnológicas acendem discussões sobre a aproximação desses dois. Técnicas novas e de mais alto nível surgem periodicamente, e são cíclicos os questionamentos quanto à validade delas, bem como a preocupação de mudanças drásticas na profissão.

Um mixer automático de música eletrônica é uma evolução natural conseguinte às tecnologias existentes. Surge da necessidade real de querer poder experimentar duas músicas juntas, para ver se casam, de forma rápida e fácil, sem precisar de um equipamento para mixá-las.

Não é uma pretensão equiparar DJ e playlist, nem substituir um pelo outro, mas sim propor mais uma tecnologia de mais alto nível que contribua, principalmente, no experimento e preparo de um DJ set. De quebra, pode aprimorar a experiência pessoal de ouvir uma playlist.

Propõe-se neste trabalho a construção de um mixer automático de música eletrônica, num algoritmo que execute os passos básicos de uma mixagem, reproduzindo, até certo ponto, o que faz um DJ. O escopo se restringe à parte mais técnica de uma mixagem e não se propõe à parte mais pessoal da discotecagem: a escolha da próxima música.

A. Contexto histórico

Disc jockeys, abreviadamente chamados de *DJs*, têm forte presença na cultura e no entretenimento mundiais desde suas origens. Termo cunhado em 1935 pelo locutor de rádio Walter Winchill, foi a forma encontrada por ele para descrever o colega Martin Block. Em seu show de rádio *Make Believe Ballroom*, Block tocava músicas dançantes e populares da época, com vinis, a fim de emular a experiência de um *ballroom*.

A partir de 1922, os primeiros DJs ganharam tanta força quanto o rádio. Nos Estados Unidos, ele se espalhava rápido como entretenimento de massas, concedendo poder e influência aos *disc jockeys*. Tanta notoriedade também trouxe oposição de toda a indústria musical: viam aquela nova forma de performance como ameaça. Martin Block foi o primeiro grande expoente dessa arte: com tanta influência em seu show, era capaz de vender qualquer produto e de levar ao topo qualquer música que tocasse.

Desde então, a figura do DJ passou por grandes transformações: hoje, é elemento essencial em qualquer evento, tem um importante papel na divulgação musical e consegue ter status de verdadeira celebridade.

II. Referencial teórico

A mixagem de uma sequência de músicas envolve mais do que tocar uma após a outra: exige a escolha adequada de músicas que combinem uma com a outra e com o momento. Independentemente de privilegiar improviso ou planejamento, um DJ precisa ser capaz de ajustar sua seleção à reação do público, realizando a chamada leitura de pista.

Além da escolha de músicas, o objetivo é tocá-las ininterruptamente com transições suaves entre elas, sobrepondo o fim de uma música ao começo da outra. Para que um DJ realize a visão sonora que tem em mente, estão à disposição nos equipamentos diversas funcionalidades, como filtros, efeitos e loops, que exigem alguma capacidade técnica para serem operadas. Em todo o processo, as possibilidades são infinitas, mas é nessa parte mais técnica que este trabalho opera.

As funcionalidades mais básicas e imprescindíveis para uma mixagem são o ajuste de BPM, os faders e os filtros de bandas de frequências. Uma forma básica de transicionar entre músicas consiste em: 1) preparar a próxima música alinhando tempo e batida; 2) começar a reproduzi-la sem som; 3) no momento apropriado, aumentar gradualmente seu volume; 4) ao mesmo tempo, abaixar as frequências graves da música anterior; 5) abaixar a música anterior. Evidentemente, nada disso é fixo e esses passos descrevem apenas uma opção de transição, dentre infinitas.

A. Literatura

Em uma das soluções já existentes na literatura, Vande Veire e De Bie (2018) propõem um sistema de discotecagem automático capaz de gerar mixagens com transições suaves em músicas de uma dada biblioteca. Baseando-se em diversas técnicas de recuperação de informação musical, o sistema analisa as estruturas musicais e leva em consideração boas práticas de discotecagem para construir uma playlist com um compromisso entre inovação e continuidade. Entretanto, o escopo fica restrito a um subgênero de música eletrônica, o *drum and bass*. Poderia ser expandido para outros subgêneros?

Em outra solução, Hirai et al. (2015) propõem um sistema semiautomático que toca músicas continuamente: o MusicMixer sugere algumas músicas para o usuário escolher a próxima, e a mixagem é feita através de um método baseado em cálculos de similaridade. A abordagem é parecida com a de Lin et al. (2009), mas esta última ainda escolhe os áudios a serem mixados dentro de uma biblioteca. Além disso, foram conduzidas avaliações subjetivas quanto à qualidade e aceitação dos resultados, que são um fator importante ao sucesso de um algoritmo como este.

III. Resultados

Nos planos iniciais, a interface de entrada era idealizada como uma lista de duas ou mais músicas. Ao longo do desenvolvimento, foram extraídas as classes Track e Mixer, esta última contendo dois objetos Track. Assim, essa passou a ser a estrutura de entrada: um par de músicas, e não mais uma lista potencialmente maior. Todavia, essa maneira é facilmente expansível para comportar qualquer número de músicas, apenas repetindo o processo de mixagem. Essas duas classes dividiram as responsabilidades dentre os processos enumerados a seguir:

A. *Análise rítmica*

O processo começa com a análise rítmica das músicas. Para identificar (i) a localização das batidas e (ii) o tempo em BPM, duas bibliotecas foram experimentadas: *librosa* e *Essentia*. Nessa análise, ambas cometiam com frequência erros de fase, i.e., batidas detectadas no contratempo ou *offbeat*. Para melhores resultados, o algoritmo aplica um filtro *low-pass* antes da análise, que será então feita sobre os graves da música. Essa medida reduz a confusão entre *kick* e outros elementos bem marcados, mas não resolve completamente a detecção deslocada de *beatgrid*.

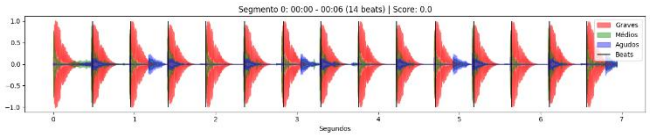

Fig. 1. Um *beatgrid* corretamente detectado.

B. *Segmentação baseada em novidade*

Momentos de maiores mudanças nas músicas são bons para começar e para terminar uma transição. Para identificá-los, foi feita uma segmentação em frases musicais numa abordagem com matrizes de autossimilaridade, como proposto por Müller (2015, C. 4).

A matriz de autossimilaridade ou *self-similarity matrix* ou SSM compara todos os momentos da música com eles mesmos e calcula a semelhança entre os elementos de cada par. Decorrente disso, a diagonal principal da SSM tem máxima similaridade, já que é a comparação de cada *frame* consigo mesmo, enquanto que sequências de momentos similares geram blocos quadrados de alta similaridade. A métrica é calculada sobre coeficientes MFCC (*Mel-Frequency Cepstral Coefficients)* normalizados e suavizados. Pontos de virada quebram sequências de alta autossimilaridade e representam limites de segmentos.

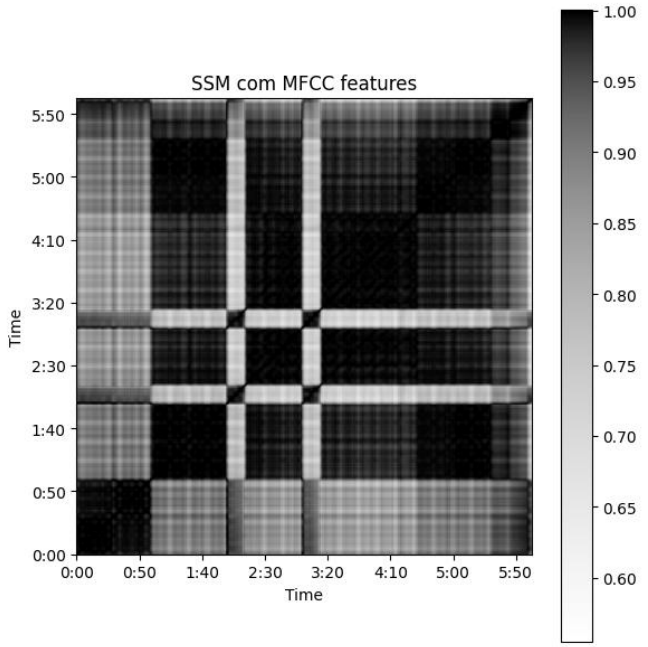

Fig. 2. Matriz de autossimilaridade da música *Surpracid*, de *Sahm*.

Para detectar os pontos de maior mudança ao longo do tempo, é calculada a diferença entre linhas consecutivas da SSM, resultando numa curva de novidade que destaca mudanças significativas nas características capturadas pelos coeficientes MFCC. Na Figura 2, pode-se ver que os limites dos segmentos na SSM correspondem aos pontos de quebra na forma de onda da Figura 3.

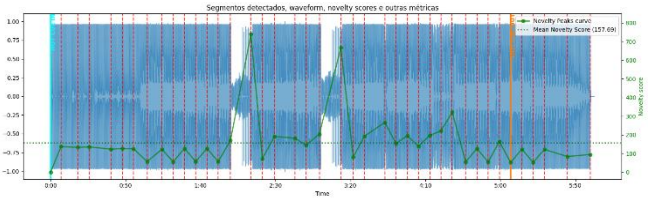

Fig. 3. Forma de onda segmentada, picos de novidade e *hot cues*.

O cálculo de picos de novidade depende de um tamanho mínimo para cada segmento. Idealmente, para o fim deste trabalho, as músicas seriam divididas numa macroestrutura: segmentos maiores que usualmente seguem o formato de *intro*, *break* 1, *drop* 1, *break* 2, *drop* 2 e *outro*. Porém, ao definir um tamanho mínimo muito grande para os segmentos, estes passam a se desencontrar de momentos importantes como os *drops*. Sendo assim, houve a preferência de segmentos mais granulares, por default com, no mínimo, 8 batidas, para evitar esse desencontro.

No entanto, essa escolha, ao melhorar o *recall*, diminui a precisão, dificultando a seleção de um segmento de entrada e de saída para a música. Assim, foi adotada uma lógica que encontra o maior pico de novidade no primeiro quinto da música e seleciona seu predecessor para ser o ponto de entrada dela. Similarmente, encontra o maior pico de novidade no último quinto da música e seleciona seu sucessor para ser o ponto de saída. Esses pontos detectados estão, respectivamente, no *hot cue in* e *hot cue out* no exemplo da Figura 3.

C. *Ajuste de tempo*

Para mixar duas músicas, é necessário ajustá-las à mesma velocidade em BPM. Alterar demais o tempo de uma música, seja esticando ou comprimindo em demasia sua forma de onda, pode causar certa descaracterização. Para evitar essas situações, o algoritmo distribui o ajuste entre as duas músicas, em vez de deixar a cargo de uma só. Assim, o BPM alvo é definido como a média dos dois BPMs e aplica-se uma taxa de *stretching* às duas músicas.

Entretanto, a música em reprodução não pode ter seu BPM alterado de uma vez, ou seria desagradável aos ouvidos. Por isso, seu ajuste é feito gradualmente numa rampa de BPM, dividindo a taxa de ajuste em pedacinhos ao longo de parametrizáveis 32 batidas, logo antes da transição. A música a entrar começa já totalmente ajustada ao BPM alvo.

À parte essa lógica, foram testados diferentes algoritmos de ajuste de tempo: os das bibliotecas *librosa*, *rubberband* e *SoX*. Como exemplo, a Figura 4 contém o espectrograma original de uma música para comparação com as figuras seguintes, que contêm os espectrogramas da mesma música após passar pelos diferentes métodos de ajuste de tempo:

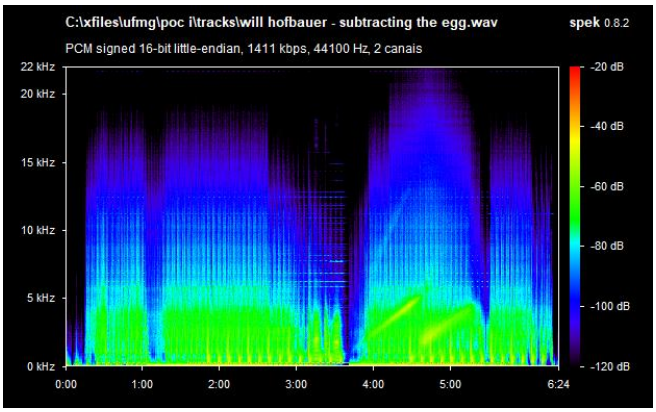

Fig. 4. Espectrograma original de *Will Hofbauer - Subtracting the egg*

librosa: utiliza *phase vocoder* e não consegue manter a qualidade de um áudio, reduzindo-a drasticamente e de forma muito perceptível aos ouvidos;

rubberband: consegue uma qualidade melhor que *librosa*, mas ainda muito ruim, efetivamente cortando qualquer frequência acima de 11kHz, como visto na Figura 5;

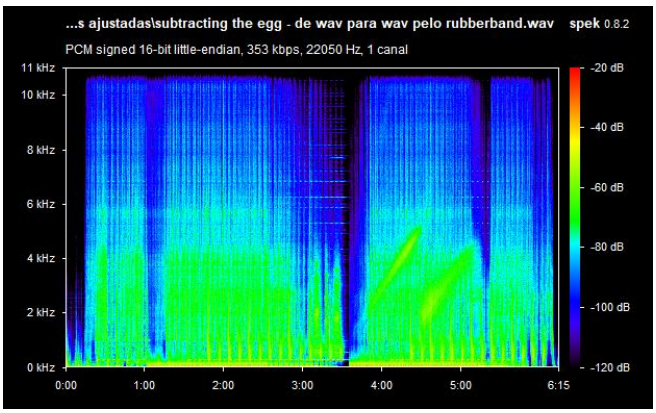

Fig. 5. Espectrograma da mesma música após passar pelo método *time_stretch* da biblioteca *rubberband.*

SoX: consegue manter a qualidade original do áudio sem alterações perceptíveis. Portanto, foi a biblioteca escolhida para este trabalho e persiste no algoritmo. Na Figura 6, o espectrograma da mesma música após passar pelo *SoX*, que permanece visivelmente igual ao espectrograma original.

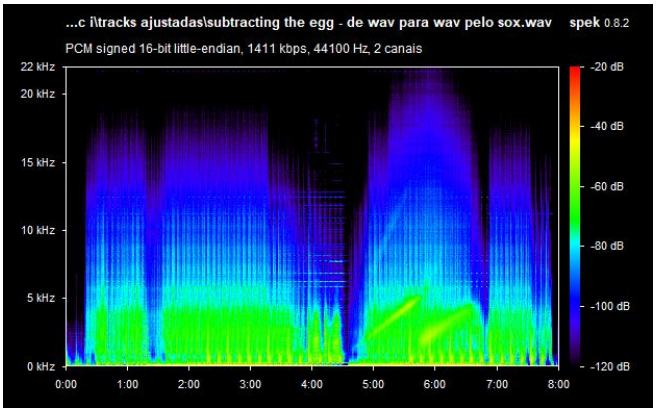

Fig. 6. Espectrograma da mesma música após passar pelo time stretching do SoX

Assim, o ajuste de tempo pode ser feito com mínima distorção, garantindo que ambas as músicas sejam sincronizadas sem comprometer sua integridade acústica. Esse processo é fundamental para a próxima etapa do pipeline, o *beatmatching*, uma vez que o alinhamento das batidas depende das velocidades das faixas.

D. *Beatmatching dinâmico*

O *beatmatching* consiste na sincronização em fase das batidas das músicas previamente ajustadas em tempo. A priori, era esperado que, alinhando a primeira batida de cada faixa, as batidas subsequentes continuassem em sincronia. Porém, não é o que acontece, devido a pequenos desalinhos em cada *beat* que se acumulam e desarranjam os posteriores, perdendo, então, a sincronia.

Assim, foi implementado um *beatmatching* dinâmico que separa o áudio em cada batida e estica ou comprime individualmente a duração desses intervalos, para atingirem a duração ideal de 60/X segundos por batida, sendo *X* o BPM alvo. Essa abordagem insere pequenos artefatos no início e no fim de cada transição que ainda carecem de serem mitigados, todavia consegue manter a sincronia das batidas.

E. *Transições*

Com todos esses alinhamentos feitos e segmentos selecionados, pode-se então partir para a transição em si. Três tipos de transição foram implementados, mas o algoritmo pode ser estendido com outros. O primeiro tipo foi um *crossfade* linear, que gradualmente abaixa uma música e aumenta a outra. Por ser linear, há uma perda de potência durante a transição nos pontos em que os sinais das duas músicas somam menos que 1. Assim, uma melhoria está no segundo tipo de transição: um *crossfade* logarítmico, que consegue manter a soma dos sinais sempre totalizando 1, como mostra a Figura 7.

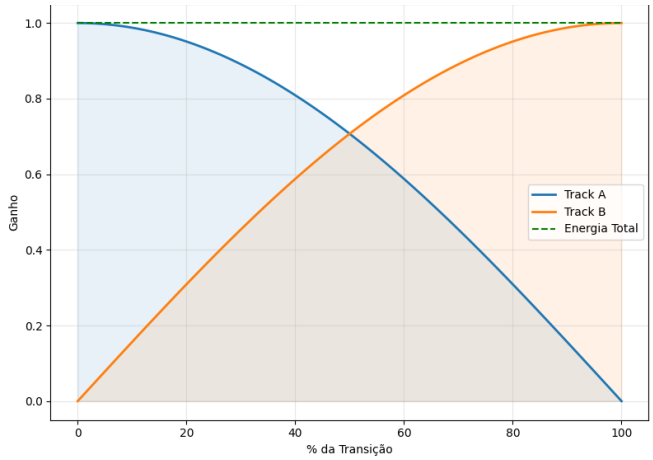

Fig. 7. Transição com um *crossfade* logarítmico.

O terceiro tipo de transição implementado executa uma troca abrupta de graves. Mais complexo, separa a música em bandas de frequências graves e médio-altas. Nestas, executa o mesmo crossfade logarítmico; naquelas, troca completamente o sinal de uma música pelo da outra no meio da transição.

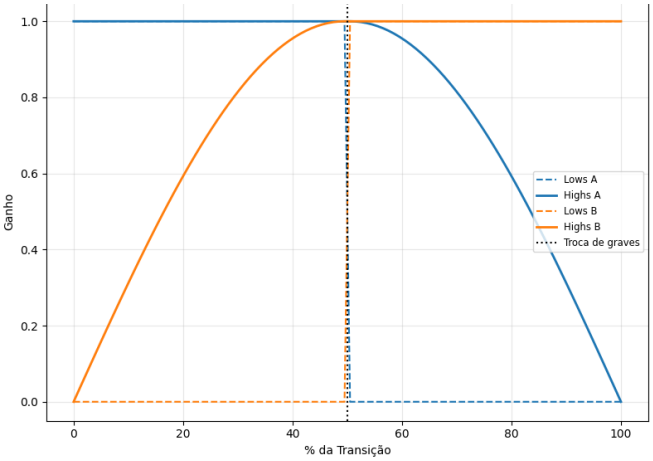

Fig. 8. Transição com troca abrupta de graves.

A troca abrupta de graves pode causar maior impacto ao ouvinte e, sendo assim, exige melhores condições de *beatmatching* e boas seleções de *hot cues in* e *out* para atingir um resultado satisfatório. Por outro lado, sendo mais simples e suave, a transição por *crossfade* logarítmico atinge melhores resultados mais facilmente, mesmo em condições de sincronia não tão ideais.

IV. Conclusão

A base de todo o algoritmo é a análise rítmica. Os algoritmos experimentados para ela marcam com frequência as batidas deslocadas. A segmentação baseada em novidade por vezes não coincide com a macroestrutura das músicas, mas funciona muito bem para segmentos de maior granularidade. A escolha de segmentos para *hot cues* também se mostrou efetiva e produz resultados satisfatórios. O *beatmatching* dinâmico consegue sincronizar as batidas, porém insere pequenos artefatos no áudio, carecendo de maior refinamento técnico. Assim, para músicas com o *beatgrid* corretamente detectado, as transições ficam boas.

Um mixer automático de música eletrônica minimamente funcional mostrou-se acessível com poucos recursos computacionais. Com um pouco mais de desenvolvimento, é possível melhorá-lo e transformá-lo num plugin para reprodutores de música, por exemplo. Assim, DJs podem se beneficiar desta solução para experimentar músicas na preparação de um set, e outras pessoas teriam a experiência melhorada de ouvir uma playlist.

Referências

[1] BREWSTER, Bill; BROUGHTON, Frank. Last night a DJ saved my life. Disponível em: <https://archive.nytimes.com/www.nytimes.com/books/first/b/brewster-dj.html>.

[2] Variety Publishing Company. Variety, p. 36. 13 de agosto de 1941. Disponível em: <https://archive.org/details/variety143-1941-08/page/n91/mode/2up>

[3] KRUH, David. Was This Man the World's First DJ?. 2019. EngineerZone: <https://ez.analog.com/ez-blogs/b/engineering-mind/posts/was-this-man-the-worlds-first-dj>. Acesso em 15/09/2023.

[4] VANDE VEIRE, L.; DE BIE, T. From raw audio to a seamless mix: creating an automated DJ system for Drum and Bass. J AUDIO SPEECH MUSIC PROC. 2018, 13 (2018). <https://doi.org/10.1186/s13636-018-0134-8>.

[5] HIRAI, Tatsunori; DOI, Hironori; MORISHIMA, Shigeo. 2015. MusicMixer: computer-aided DJ system based on an automatic song mixing. In Proceedings of the 12th International Conference on Advances in Computer Entertainment Technology (ACE '15). Association for Computing Machinery, New York, NY, USA, Article 41, 1–5. <https://doi.org/10.1145/2832932.2832942>.

[6] LIN, Heng-Yi; LIN, Yin-Tzu; HU, Min-Chun; WU, Ja-Ling. 2009. *Music Paste: Concatenating Music Clips based on Chroma and Rhythm Features.* In Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009). 213-218. <https://archives.ismir.net/ismir2009/paper/000073.pdf>

[7] MÜLLER, Meinard. 2015. Fundamentals of Music Processing. Chapters 3 and 4.

[8] MÜLLER, Meinard; ZALKOW, Frank. libfmp: A Python Package for Fundamentals of Music Processing. The Journal of Open Source Software (JOSS), 6(63), 2021.

[9] MÜLLER, Meinard; ZALKOW, Frank. FMP Notebooks: Educational Material for Teaching and Learning Fundamentals of Music Processing. Proceedings of the International Conference on Music Information Retrieval (ISMIR), Delft, The Netherlands, 2019: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C0/C0.html

[10] LIBROSA. librosa.effects.time_stretch. Acesso em: 27 jan. 2025. Disponível em: <https://librosa.org/doc/latest/generated/librosa.effects.time_stretch.html>.

[11] PYRUBBERBAND. pyrubberband.pyrb.time_stretch. Disponível em: <https://pyrubberband.readthedocs.io/en/stable/generated/pyrubberband.pyrb.time_stretch.html>. Acesso em: 27 jan. 2025.

[12] GITHUB. SoX - Sound eXchange. Disponível em: <https://github.com/chirlu/sox>. Acesso em: 27 jan. 2025.