Gestus Magni -Sinfonia da Expressão Corporal

Lucas Martins – 79294 – a79294@ualg.pt

Pedro Gonçalves – 79297 – a79297@ualg.pt

Relatório Técnico – LESTI – Computação Visual

Resumo

Este trabalho tem como objetivo desenvolver uma solução para a criação de música de maneira interativa utilizando tecnologia de deteção de movimento, de faces e de mãos. A solução será realizada com interação de um programa desenvolvido em *Python* e o programa *Blender*, onde são capturados e analisados movimentos e gestos de um utilizador, de forma a traduzi-los em notas musicais, criando uma interface inovadora de composição musical. O problema em estudo é a falta de interfaces intuitivas que conectem o movimento humano diretamente à música, limitando a expressão criativa no mundo da música. Este é um tema relevante, pois muitos artistas não possuem os meios para produzir sons por meios tradicionais, seja devido a limitações físicas, financeiras ou até mesmo por falta de acesso a recursos como os próprios instrumentos, e com esta solução pretendemos fornecer uma alternativa tecnológica para explorar novas formas de expressão.

Esta abordagem é eficaz, pois oferece uma interação natural entre um utilizador e a música, promovendo uma experiência imersiva e acessível. Além disso, permite uma adaptação da linguagem gestual em tempo real, garantindo flexibilidade na criação musical.

1. Introdução

A relação entre movimento humano e música tem sido explorada ao longo dos anos em diferentes contextos, desde atuações ao vivo a sistemas de composição assistida por computadores. Um dos principais desafios deste campo é a criação de interfaces intuitivas que permitam a utilizadores interagir diretamente com a música por meio do movimento. Em muitos casos, os sistemas que existem dependem de dispositivos complexos ou caros, como controladores/teclados ***MIDI*** ou sensores específicos, o que pode ser um obstáculo para muitos artistas.

Este trabalho pretende responder à questão de como é possível desenvolver uma interface interativa que traduza gestos e movimentos corporais em notas musicais, permitindo uma nova forma de criação de musical acessível e intuitiva.

1.1. Inspiração – Artista '*Imogen Heap*'

A cantora-compositora britânica *Imogen Heap* foi uma grande fonte de inspiração para este trabalho devido à sua abordagem inovadora e experimental na fusão de tecnologia e música. *Heap* é reconhecida mundialmente pelo uso de dispositivos eletrônicos e gestuais para manipular sons em tempo real, criando atuações imersivas e interativas.

Um dos seus grandes feitos foi a criação das luvas "*Mi.Mu*" – umas luvas com sensores que permitem a manipulação de música através de movimentos das mãos. Essa tecnologia permitiu à artista controlar aspetos da música, como efeitos (*e.g* eco*, reverb)*, volume, *pitch*, apenas com gestos corporais, revolucionando a forma como a música pode ser criada e experienciada ao vivo.

Este trabalho pretende remover a necessidade de um dispositivo externo para o mesmo efeito, permitindo desenvolver peças de música utilizando apenas um computador com uma camara e tecnologia de ***machine-learning***[2].

Índice

2. Execução do Programa – Requisitos e Manual de Instruções

Para melhor entendimento das condições necessárias para executar o programa e como o executar, recomendamos que consultem o ficheiro "READ.ME"

3. Funcionalidades e Comportamento do Programa

Como referido previamente, o tema do trabalho em questão é a criação de música através da interação de partes do corpo. Neste capítulo, serão exploradas as funcionalidades do programa; O código principal foi desenvolvido na linguagem de Python, utilizando a ferramenta de desenvolvimento colaborativo *Github* para que vários utilizadores possa ter a capacidade de contribuir para o projeto simultaneamento.

3.1. Reconhecimento de Linguagem Gestual Americana (ASL) e Mãos

Adaptado do repositório '*sign-language-detector-flask-python'* do projeto de deteção de Linguagem Gestual Americana do utilizador 'SohamPrajapati', utilizamos a solução de computação visual "*Mediapipe"* para detetar mãos e por sua vez os dedos, podendo destacar e utilizar vários pontos da mão e dedos, por exemplo, a ponta do dedo indicador, de maneira a conseguir interagir com o programa. Estes pontos são chamados de *landmarks*.
Sendo possível detetar as mãos e *landmarks*, esta informação é passada pelo algoritmo do modelo pré-treinado de *machine-learning*, e por sua vez, ser possível detetar diferentes letras de ASL.
Para treinar o modelo de maneira que o mesmo consiga interpretar as diferentes letras, são tiradas um conjunto de fotos que representam uma dada letra, fazendo isto para cada letra, temos então um *dataset.* Este *dataset* é então passado por um algoritmo '*Random Forest'*, e desta maneira o modelo chega a uma conclusão para interpretar as diferentes letras.

Para este projeto foi usado a linguagem gestual americana, mas alterando o *dataset* e treinando o modelo de outra forma, o programa conseguiria funcionar para qualquer tipo de linguagem gestual.

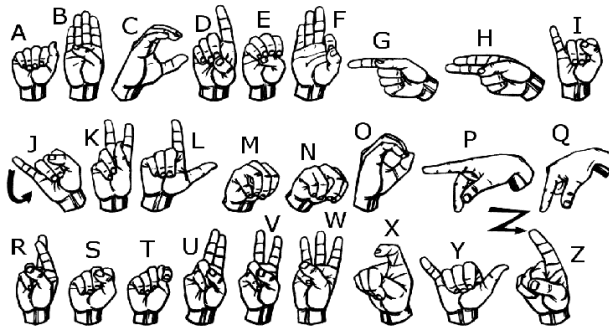

Figura 1. Representação das letras em ASL

3.2. Reprodução de Música

Executando o programa e fazendo os gestos que representam as diferentes letras, temos 5 letras diferentes que representam 5 sons diferentes.
Para este projeto, acabamos por considerar utilizar apenas 5 letras por uma questão de coerência. Por exemplo, supondo que existisse um som para cada letra e apresentássemos primeiro a mão da forma da letra B e de seguida a letra O; na fase de transição das letras, existe a possibilidade do modelo detetar a letra C, levando a reprodução de um som indesejado. Desta forma as 5 letras que consideramos são bastantes distintas em aspeto e são as seguintes:

"A", "B", "C", "D" e "I"

Uma forma de remediar esta questão seria incluir um tempo de atraso de um certo tempo entra a deteção de duas letras, mas aí estaríamos a limitar o fluxo/ritmo da música. A solução é realizar os gestos de maneira rápida, impedindo que o modelo detete a letra indesejada.

Os diferentes sons são reproduzidos através da biblioteca *pygame,* carregando os ficheiros dos sons e utilizando a função '*play()*' para reproduzir o som.

Os sons são associados a uma letra através da estrutura de dados em *Python* conhecida como dicionários, tendo cada letra a apontar para o diretório de uma nota.

3.2.1. Instrumentos

Por defeito, o instrumento a ser tocado é o piano, onde as letras representam os seguintes sons:

LETRA	SOM
A	Dó
B	Ré
C	Mi
D	Fá
I	Sol

Não houve nenhum critério de escolha de som para uma dada letra, tanto para o piano como para qualquer outro instrumento.

Se desejar pausar alguma música/som a vir de um instrumento, apresente a letra "U"

3.2.2. Reconhecimento de Objetos

O programa tem a inclusão do modelo de deteção de objetos YOLOv11 (*You Only Look Once*), cortesia da empresa *Ultralytics*, que permite detetar uma variadede de objetos.

Para o contexto do trabalho, adaptamos o YOLO de maneira a poder alterar entre os diferentes instrumentos a ser tocados, ou seja, se um dado objeto é detetado, um instrumento é selecionado, e funciona da seguinte forma; no terminal onde o programa está a ser executado (terminal do VSCode, linha de comandos, etc) se escrever 'obj' e pressionar a tecla *Enter*, o modelo de objetos começa a ser executado e consoante a tabela de baixo, o intrumento/funcionalidade é selecionada:

Objeto a Apresentar	Objeto Detetado pelo YOLO	Instrumento/Funcionalidade selecionada
Garrafa	*Bottle*	Bongo
Telemóvel	*Phone*	Piano
Planta num vaso	*Potted Plant*	Bateria
Copo	*Cup*	Música Aleatoria
Mochila	*Backpack*	Música de Fundo

É de notar que o som do instrumento associado a uma letra está definido no código e facilmente se

pode fazer com que a letra A em vez de tocar a nota 'Dó' que toque 'Ré'.

Os sons dos intrumentos foram obtidos através do *website "Freesound*", que disponibiliza milhares de *packs* de sons, sejam instrumentos ou sons de temas aleatórios. Os ficheiros dos sons estão localizados nas pastas "bongo", "drums", "metronome", "piano" e "custom_music".

Para parar o YOLO, deve escrever 'stop' e pressionar a tecla *Enter* no terminal.

3.2.3. Funcionalidade da Mochila e Copo – Música de Fundo

A deteteção de uma mochila no decorrer da execução do YOLO permite ao utilizador que interage com o programa reproduzir uma música de fundo da pasta "custom_music" através de uma janela que aparece quando a mochila é detetada.

Esta funcionalidade foi desenvolvido utilizando a biblioteca Tkinter – a biblioteca nativa ao *Python* de desenvolvimento de interfaces gráficas.

Apresentando um copo é selecionada aleatoriamente uma música da mesma pasta referida anteriormente.

3.3. Reconhecimento de Faces

O programa também toma proveito do *Mediapipe* para reconhecer faces e utiliza-as para interagir com o programa.
Da mesma maneira que as mãos têm *landmarks*, o *Mediapipe* também consegue reconhecer faces e obter *landmarks*, fazendo subdivisões de uma faces em pequenos triângulos.
O programa utiliza esta capacidade para subir ou descer um octave do piano, se este for o instrumento selecionado no momento.
Caso a face esteja inclinada para cima ou para baixo, a nota do piano sobe ou desce, respetivamente, um octavo.
Os *landmarks* correspondem a um array onde cada elemento é traduzido num vértice da face. Utilizando estes *landmarks*, conseguimos calcular a inclinação de uma face, e por sua vez usar esta para subir um octavo a nota do piano, caso a face esteja a olhar para cima e descer um octavo da nota do piano caso a face esteja a olhar para baixo.

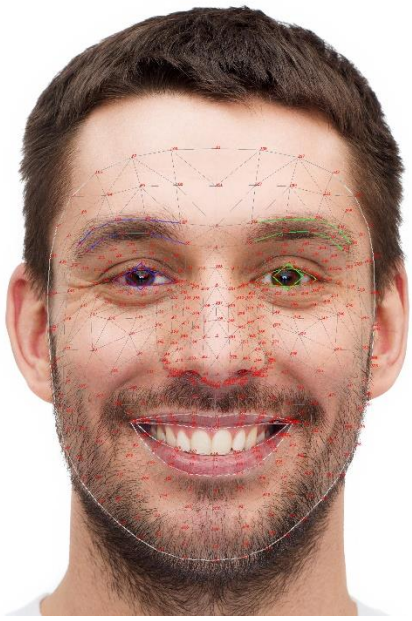

Figura 2. Elementos dos *landmarks* da face (*Mediapipe*)

Sabendo que a ponta do nariz corresponde ao 1º, o ponto mais baixo do queixo ao 152º e o ponto mais alto testa ao 10º elemento, conseguimos calcular a inclinação da seguinte forma:
- Altura e largura da janela da webcam (*frame)* é obtida.
- Multiplicamos a coordenada na vertical (y) da ponta do nariz , do queixo e da testa pela a altura do *frame*, para que tenhamos valores consistentes, independentemente do tamanho do *frame*.
- Definimos que a altura da face seja o valor absoluto, ou seja, positivo, da diferença entre o '*y*' da testa e o do queixo. É de notar que o '*y*' começa no topo do *frame*, ou seja, ao contrário do eixo '*y*' de um referencial cartesiano, portanto técnicamente a distância vertical da testa é menor que a do queixo.
-Tendo a altura da face, concluimos que a inclinação da mesma é calculada fazendo a diferença entre a altura do queixo e a altura do nariz, dividindo pela altura da face, para termos um valor proporcial, independentemente da proximidade da face à câmera.

Para simplificar, a formula da inclinação pode ser representada da seguinte forma:

$$\frac{(queixo_y * altura_{frame}) - (nariz_y * altura_{frame})}{\text{Valor Positivo}((testa_y * altura_{frame}) - (queixo_y * altura_{frame}))}$$

Para responder à questão do "porque é que se utiliza a formula para determinar a inclinação?", analisemos os casos das seguintes fotos abaixo:

Observando as barras nas imagens, que representam a altura do queixo ao à ponta do nariz, conseguimos perceber que, visto de frente, quanto menor for a altura, mais para baixo será a inclinação da cabeça e vice-versa.

Após analisarmos os valores obtidos executando a formula num programa de *Python*, definimos *thresholds*, isto é, limites, e que caso o valor obtido seja maior que 0.5 a cabeça está para cima, tocando notas com um octavo superior, e se o valor for menor que 0.4 a cabeça está para baixo, tocando notas com um octavo inferior. Se o valor for menor que 0.5 e maior que 0.4, a cabeça está neutra, tocando as notas no seu valor '*default*' .

Os sons com o *pitch* (*i.e* octave) alterado foram obtidos passando os ficheiros originais do piano pelo o *website* https://vocalremover.org/pitch.

3.4. Menu de Ajuda

Para ajudar o utilizador a entender melhor o funcionamento do programa, incluimos um menu de ajuda. O programa continuamente lê a posição da ponta do dedo indicador e se esta estiver por cima do ponto de interrogação localizado no canto superior direito do *frame*, uma imagem com uma breve explicação do funcionamento do programa aparece.

3.5. Metrónomo

O programa dispõe de uma funcionalidade de Metrónomo – um aparelho que através de pulsos de duração regular, indica um andamento musical. Se duas mãos forem apresentadas no *frame*, o metrónomo comçea a pulsar a um ritmo de 100 batimentos por minuto. Apresentando novamente duas mãos, o metrónomo pára. O som para esta funcionalidade também foi obtido do *website* do "*Freesound*".

3.6. Implementação do Blender

O *Blender* foi utilizado no projeto para a criação de um *visualizer* em tempo real dos dados de áudio recebidos pelo programa principal do *Python*, criando uma representação visual dinâmica do sons produzidos.

3.6.1. Arquitetura do Sistema

O *visualizer* opera através de uma conexão cliente-servidor local, utilizando comunicação TCP/IP através da porta 65432 em ***localhost*** **(127.0.0.1)**. Para garantir a robustez do sistema, foi implementado um mecanismo de reconexão automática que monitoriza constantemente o estado de conexão e restabelece a comunicação em caso de falhas.

3.6.2. Componentes Visuais

O sistema visual é composto por 32 cubos distintos, onde cada um representa uma faixa específica de frequência de áudio. Os cubos, são identificados sequencialmente de "**Cube.000**" até "**Cube.031**" e têm uma altura que é alterada dinamicamente baseado na amplitude da frequência correspondente.

A implementação inclui um objeto de luz denominado *'light-fly'*, cuja intensidade varia de acordo com a amplitude geral do áudio. Através de um shader de emissão, é calcula a intensidade da luz utilizando a fórmula

$$(\text{amplitude média} * 2.0) + 1.0$$

criando um ambiente visual dinâmico que responde às variações sonoras.

3.6.3. Processamento de Dados

O processamento das frequências de áudio é realizado através de cinco categorias principais, cada uma com sua própria amplificação: graves (0-20%) com amplificação 4.0x, médios-graves (20-40%) com 3.0x, médios (40-60%) com 2.5x, médios-agudos (60-80%) com 2.0x e agudos (80-100%) com 1.5x.

Para garantir uma visualização fluida e natural, foi implementado um sistema de interpolação suave de 30% entre os estados dos cubos. Um limite mínimo de altura de 0.1 unidades foi estabelecido para manter a visibilidade constante, enquanto uma variação aleatória de ±10% adiciona um movimento orgânico às animações dos cubos. O sistema atualiza-se em tempo real, mantendo sincronização com os frames do Blender.

3.6.4. Sistema Multi-thread / Otimizações

O *visualizer* utiliza um sistema *multi-thread*, onde a thread principal é responsável pela gestão da animação do Blender, enquanto uma *thread* secundária gerencia a receção e processamento dos dados de áudio. A comunicação entre threads é realizada através de filas *thread-safe*, e o estado da conexão é controlado por um sistema de eventos.

O sistema incorpora várias otimizações para garantir um desempenho eficiente. O processamento é realizado utilizando *arrays* da biblioteca de *Python "**numpy**"* para a manipulação eficiente dos dados, complementado por um sistema de *buffer* para a gestão do áudio. Além disso, implementou-se um mecanismo de recuperação automática de conexão e controles para prevenir distorções visuais através de limitações de altura máxima.

3.6.5. Integração com Programa Principal

A integração entre o Blender e o programa principal é realizado através de uma comunicação bidirecional, permitindo sincronização automática com os dados de áudio.

Esta implementação no Blender funciona como um complemento visual essencial ao sistema principal, proporcionando feedback em tempo real das interações musicais do utilizador. A combinação de elementos visuais dinâmicos com a representação sonora cria uma experiência mais completa e envolvente para o utilizador final.

4. Conclusão

Este trabalho apresentou uma solução inovadora para a criação interativa de música, utilizando tecnologias de detecção de movimento, reconhecimento de mãos e faces, e machine learning. O objetivo foi criar uma interface acessível, permitindo que gestos corporais sejam convertidos em notas musicais, proporcionando uma nova forma de composição sem a necessidade de instrumentos tradicionais.

O program, desenvolvido em *Python* e utilizando técnologias como *Mediapipe* e *Blender*, transforma gestos de linguagem gestual americana (ASL) em sons, e também adapta o pitch das notas de acordo com a posição da face. Além disso, a deteção de objetos permite alternar entre instrumentos musicais, tornando a interação mais dinâmica.

Apesar de eficaz, o sistema enfrenta desafios, como a precisão na deteção de gestos e a possível geração de sons indesejados durante transições rápidas de gestos.

Em resumo, este projeto demonstra o potencial das tecnologias de visão computacional e interação gestual para criar uma experiência musical interativa, acessível e inovadora, abrindo novas possibilidades para a expressão artística e a criação musical

Referências

1. MIDI - https://pt.wikipedia.org/wiki/MIDI
2. Imogen Heap | Mi.Mu Gloves - https://youtu.be/3QtklTXbKUQ?si=C2M1AUM2bx_-Efz3&t=561
3. Projeto ASL - https://github.com/SohamPrajapati/sign-language-detector-flask-python
4. MediaPipe; Elementos *landmark* - https://ai.google.dev/edge/mediapipe/solut

ions/vision/face_landmarker#configurations_options
5. Metrónomo - https://pt.wikipedia.org/wiki/Metr%C3%B4nomo
6. Alteração de Pitch - *https://vocalremover.org/pitch*.
7. Instrumentos e metrónomo - https://freesound.org/