Geração de expressões faciais por emoção e fala

Facial expression generation through speech and emotion

Arthur William Dórea Melo[1], Caio Vasconcelos Araújo Figueiredo[1], Pedro Moreira Guerra de Almeida[1], Renan Silva Ferreira[1], Guilherme Vieira Moraes[1], Victor Flávio de Andrade Araujo[1,2]

[1]Universidade Tiradentes (UNIT), Aracaju, SE – Brasil
[2] National Institute of Science and Technology Social and Affective Neuroscience (INCT-SANI)

(arthur.william, caio.varaujo, pedro.guerra, renan.ferreira

guilherme.vmoraes, victor.flavio93)@souunit.com.br

Abstract. Introduction: *This article presents a method to convert audio files into facial expressions.* ***Objective:*** *Conversion of emotional audios into matching facial expressions.* ***Methodology or Steps:*** *This article uses interpolation, C#, Unity, Google Colab and Adobe Firefly to do the audio-to-face-expression conversions.* ***Results:*** *Successful conversions of audio files to equivalent facial expressions.*
Keywords *AI, audio-to-image, Unity, Face, Emotion.*

Resumo. Introdução: *Este artigo apresenta um método de converter arquivos de áudio em expressões faciais.* ***Objetivo:*** *Transformação de áudios emotivos em expressões faciais correspondentes.* ***Metodologia ou Etapas:*** *Este trabalho utiliza interpolação, C#, Unity, Google Colab e Adobe Firefly para realizar as conversões de áudio para expressões faciais.* ***Resultados:*** *Conversões bem sucedidas de arquivos de áudio em expressões faciais equivalentes.*
Palavras-Chave *IA, áudio-para-imagem, Unity, Face, Emoção.*

1. Introdução e Justificativa

Agentes inteligentes podem ser mais convincentes caso eles apresentem uma expressão facial correspondente ao que é dito por eles [Buck et al. 1972]. Desta forma, com o rápido avanço das tecnologias de identificação de fala, é interessante que exista um sistema que converta tais falas em expressões faciais correspondentes para melhorar a performance da comunicação [Buck et al. 1972]. Este artigo tem como objetivo a criação de um sistema desse tipo, utilizando métodos como *SVM (Support Vector Machines)* e *hmm(Hidden Markov Model)* [Kwon et al. 2003] para estipular níveis de emoções e aplicá-los a um rosto por *prompt* [Wu et al. 2025] ou interpolação [Zhang 1999].

Este sistema é particularmente interessante para desenvolvedores que buscam uma melhor comunicação entre agentes inteligentes e usuários em situações como atendimentos, home pages ou tradutores de língua por áudio.

2. Fundamentação teórica

Se tratando sobre a base do sistema de interpolação, ela foi inspirada em [Bassili 1979], nele são descritos detalhadamente o comportamento de pontos faciais

em diversas expressões. Para reconhecer a voz, utilizamos os métodos descritos por [Dellaert et al. 1996] e [Seehapoch e Wongthanavasu 2013], com destaque na utilização de SVM (Support Vector Machine) e Hidden Markov Models como em [Kwon et al. 2003]. Em [Wu et al. 2025] as técnicas mostradas guiam para a criação de *prompts*nos modelos de difusão, enquanto a parte para interpolação matemática foi usado [Zhang 1999] que forneceu uma base teórica.

3. Metologia

3.1. Análise de melhoria

Após a criação da célula no *Google Colab*, o grupo analisou como a melhoria poderia ser implementada. O código desenvolvido utiliza *SVM*[Milton et al. 2013], *Hidden Markov Model* e a base de dados SUSAS *database* para identificar as porcentagens de cada emoção nos arquivos de áudio [Kwon et al. 2003] [Dellaert et al. 1996],

A abordagem escolhida foi a mais simples: Exportar os dados processados após a execução do código da célula no ambiente *Colab* em forma de arquivos de texto padronizados, e então utilizar este mesmo arquivo e formato de texto na Unity para a criação das expressões faciais, para assim utilizar a saída do *Colab* como uma entrada na Unity, criando a ligação entre os dois sistemas.

Uma vez na Unity, os dados precisariam ser convertidos em expressões faciais, e nessa etapa houve uma divergência: Utilizar interpolação matemática [Zhang 1999] para criar uma face procedural baseada em outros modelos de face predeterminados pelo grupo, ou usar modelos de difusão para criação de uma imagem por *prompt*[Wu et al. 2025].

3.1.1. Utilizando interpolação

Esta foi a primeira solução a ser pensada, essencialmente a ideia é criar um conjunto de expressões faciais predeterminadas, que por sua vez são conjuntos de pontos faciais, e interpolar entre elas conforme o valor das expressões resultantes do código do *Colab*. Dessa forma, seria possível gerar expressões procedurais instantaneamente.

Os pontos positivos desse método são: Não é necessária conexão à internet nem um banco de dados local de larga escala, a simplicidade de implementação (que consiste em simples interpolações entre pontos), ajustes acessíveis e consistentes das expressões resultantes, podendo facilmente alterar a porcentagem de cada expressão conforme a necessidade, e geração instantânea.

3.1.2. Utilizando modelos de difusão

Esta foi a segunda solução e a que foi sugerida pelo professor: Utilizar os dados de emoções resultantes do *Colab*, para criar uma imagem em uma ferramenta de IA generativa utilizando modelos de difusão.

Os pontos positivos desse método são: As imagens em si serão mais detalhadas, o processo de criação por *prompt*facilita o desenvolvimento, e a geração de imagens utilizará um banco de dados extenso, o que reduz a estranheza [Wu et al. 2025].

Foi decidido pelo grupo a utilização dos dois métodos, de forma a analisar qual método seria mais adequado e para qual situação.

3.2. Execução utilizando interpolação

O grupo utilizou o componente *Transform* de objetos vazios na Unity para gerar pontos faciais, os pontos foram padronizados e armazenados em instâncias da classe *Face Expression*, que armazena os pontos de cada parte do rosto (olho esquerdo, sombrancelha esquerda, olho direito, sombrancelha direita e boca). As linhas que ligam os pontos de cada parte da face foram criadas utilizando o componente nativo *LineRenderer*.

Com os modelos predefinidos de expressões prontos, o grupo avançou para o sistema de interpolação entre as expressões faciais utilizando C#. Após tentativas de usar interpolações simples, o grupo optou por utilizar interpolações multivariadas, que foi mais adequado. A equação 1 demonstra a equação resultante para cada emoção.

$$P_a = \sum P_e * clamp(m_e, 0, 1) \tag{1}$$

Em que:
P_r = Posição resultante do ponto facial
P_e = Posição do ponto facial da emoção
m_e = Multiplicador percentual da emoção

Foi considerado também o caso de não existir emoção o suficiente, que neste caso prevaleceria um rosto neutro, para aplicar isto no sistema, foi considerado o neutro como sendo a escassez de emoções detectadas, o coeficiente de neutralidade foi calculado conforme a equação 2.

$$C_n = clamp(1 - \sum e_x, 0, 1) \tag{2}$$

Em que:
C_n = Coeficiente de neutralidade
e_x = Coeficiente da emoção X

Finalmente, a posição de cada ponto do rosto é denotada na equação 3.

$$P_a = \sum (P_e * clamp(m_e, 0, 1)) + P_n * clamp(1 - \sum e_x, 0, 1) \tag{3}$$

Em que:
P_a = Posição resultante do ponto facial
P_e = Posição do ponto facial da emoção
m_e = Multiplicador percentual da emoção
P_n = Posição do ponto facial neutro
e_x = Coeficiente da emoção X

É importante notar que o código da Unity criado pela equipe e que é responsável pela interpolação não é limitado à ter seu somatório dos coeficientes de emoção igual

(a) 100% neutro (modelo neutro). **(b) 50% raiva, 50% neutro.** **(c) 100% raiva (modelo raiva).** **(d) 100% feliz (modelo feliz).** **(e) Pontos da expressão feliz.** **(f) Pontos da expressão raivosa.**

Figura 1. Expressões faciais do sistema na Unity.

a 100%, já que cada emoção possui um coeficiente que varia entre 0 (0%) e 1 (100%). Isto significa que pode ser gerado um rosto com 100% de raiva e, ao mesmo, 100% de felicidade.

Uma vez implementada esta expressão no código na *Unity* para cada ponto facial do rosto resultante, o sistema de geração de expressões faciais utilizando interpolação passou a funcionar adequadamente. A Figura 1 demonstra alguns resultados das interpolações e seus respectivos coeficientes de emoção no ambiente de execução da Unity, e a Figura 1(e) e Figura 1(f) demonstra a distribuição dos pontos faciais em duas expressões no mesmo ambiente.

3.3. Execução utilizando modelos de difusão

Conforme ocorreu com o método por interpolação, o método por modelo de difusão utilizou os coeficientes de emoção dos áudios extraídos do *Colab* para gerar uma imagem, o modelo seguido para o *prompt*foi o seguinte: "Um rosto com [porcentagem da emoção]% de [emoção]".

A fim de realizar uma comparação justa, os *prompts* no *Adobe Firefly* foram adaptados de forma que os resultados fossem cartunizados. Dessa forma, eles seriam mais comparáveis com os resultados por interpolação feitos anteriormente, e reduziria o viés que poderia ser causado pelo realismo ou pela falta dele.

A Figura 2 demonstra as comparações entre os resultados obtidos por cada método, frente aos mesmos coeficientes de emoção.

Conforme mencionado por Wu [Wu et al. 2025], *prompts* mais complexos causavam perda de coerência nos resultados gerados pelos modelos de difusão, enquanto que o sistema por interpolação se manteve mais consistente frente a expressões faciais complexas. Em um determinado ponto de complexidade, o *Adobe Firefly* não conseguiu gerar imagens, o resultado por interpolação frente às mesmas condições é visível na Figura 3.

3.3.1. Comparação com soluções da indústria

Atualmente já existem ferramentas comercias como o MetaHuman Creator, da Unreal Engine 5, que seria voltada para a criação de personagens mais realistas com expressões faciais mais difíceis. Nesse trabalho foram apresentadas propostas que se diferenciam por mostrar uma alternativa de baixo custo e com maior acessibilidade, especialmente a

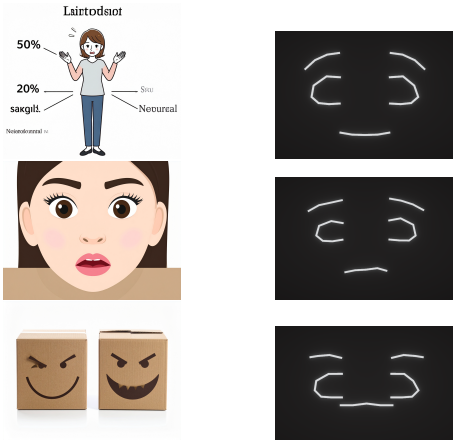

Figura 2. Resultados, por difusão à esquerda e por interpolação à direita. Linha 1: 50% felicidade, 20% surpresa, 30% neutralidade. Linha 2: 30% nojo, 20% surpresa, 50% neutralidade. Linha 3: 100% felicidade, 100% raiva.

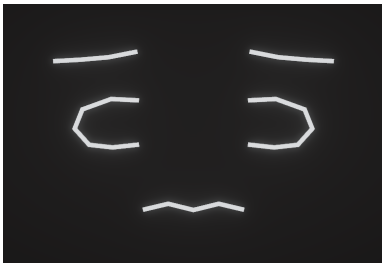

Figura 3. Rosto gerado por interpolação com 55% medo, 25% raiva, 5% tristeza e 5% neutralidade.

respeito à integração com a Unity e a possibilidade de utilização offline com geração procedural. Para aplicações educacionais, assistivas ou de prototipagem rápida essa abordagem se mostra muito promissora.

3.4. Resultados

A conversão de arquivos de áudio para expressão facial funcionou tanto utilizando o método de modelos de difusão quanto o de interpolação. O método por interpolação, no entanto, se mostrou mais consistente, enquanto que o método por difusão trouxe imagens mais detalhadas e diversificadas. Porém, em condições de alta complexidade, os modelos de difusão se tornam menos eficientes, visível na Figura 2.

4. Considerações Finais

Tanto o uso de interpolação entre expressões faciais quanto o uso de modelos de difusão são métodos válidos e efetivos para a geração de expressões faciais por voz. Em casos em que a prioridade for: Consistência, funcionamento *off-line*, capacidade de *fine tuning* e tempo de resposta, o sistema por interpolação será mais adequado. Contudo, se a prioridade for: Tridimensionalidade, realismo ou detalhes, o sistema por difusão é a melhor escolha, porém é importante considerar um banco de dados otimizado para isso, dado que, em modelos de difusão convencionais, a precisão cai drasticamente conforme a especificadade do *prompt* aumenta [Wu et al. 2025].

Agradecimentos

Este estudo foi financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil (CNPq), por meio dos Processos nº 309228/2021-2; 406463/2022-0; 153641/2024-0

Referências

Bassili, J. N. (1979). Emotion recognition: the role of facial movement and the relative importance of upper and lower areas of the face. *Journal of personality and social psychology*, 37(11):2049.

Buck, R. W., Savin, V. J., Miller, R. E., e Caul, W. F. (1972). Communication of affect through facial expressions in humans. *Journal of personality and social psychology*, 23(3):362.

Dellaert, F., Polzin, T., e Waibel, A. (1996). Recognizing emotion in speech. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 3, pages 1970–1973. IEEE.

Kwon, O.-W., Chan, K., Hao, J., e Lee, T.-W. (2003). Emotion recognition by speech signals. In *Interspeech*, pages 125–128. Citeseer.

Milton, A., Roy, S. S., e Selvi, S. T. (2013). Svm scheme for speech emotion recognition using mfcc feature. *International Journal of Computer Applications*, 69(9).

Seehapoch, T. e Wongthanavasu, S. (2013). Speech emotion recognition using support vector machines. In *2013 5th international conference on Knowledge and smart technology (KST)*, pages 86–91. IEEE.

Wu, W., Li, Z., He, Y., Shou, M. Z., Shen, C., Cheng, L., Li, Y., Gao, T., e Zhang, D. (2025). Paragraph-to-image generation with information-enriched diffusion model. *International Journal of Computer Vision*, pages 1–22.

Zhang, J. (1999). C-bézier curves and surfaces. *Graphical Models and Image Processing*, 61(1):2–15.