Controle de Navegação por Gestos: Uma Solução Baseada em Visão por Computado

Gustavo Rodrigues Bassaco
Engenharia de Computação
Universidade Tecnológica Federal do Paraná
Campus de Apucarana
gustavobassaco@alunos.utfpr.edu.br

***Abstract*—This paper presents a project aimed at creating a virtual mouse controlled by hand gestures using Python programming language. The project utilizes OpenCV and Mediapipe Hands libraries for hand tracking and gesture recognition. The system is able to map hand movements in real time and infer the gestures made by the user, allowing for control of the computer's cursor, clicking, scrolling, and zooming functions.**

***Index Terms*—artificial intelligence, gesture recognition, virtual mouse**

I. Introdução

Este artigo apresenta um projeto de desenvolvimento de um programa em Python capaz de mapear a mão em tempo real e inferir qual gesto está sendo realizado pelo usuário, com o objetivo de criar um mouse virtual para controlar o computador.

Este projeto tem como motivação a necessidade de uma interação mais intuitiva e natural com o computador, especialmente para pessoas que possuem alguma dificuldade em utilizar o mouse convencional. Além disso, o programa pode ser útil em situações em que o mouse físico não está disponível, como em apresentações ou em casos de danos ao equipamento.

A. Contextualização do projeto

A interação homem-máquina é um tema importante na área de tecnologia da informação e tem evoluído constantemente. Com o avanço da tecnologia de reconhecimento de imagem, tem sido possível criar programas que permitem a interação com o computador através de gestos corporais. O reconhecimento de gestos é um ramo da visão computacional que tem como objetivo interpretar os movimentos do corpo humano e convertê-los em comandos para controlar dispositivos eletrônicos.

O desenvolvimento de um mouse virtual que utilize a mão como dispositivo de controle é uma das aplicações possíveis deste tipo de tecnologia. Dessa forma, o usuário pode controlar o computador de forma mais intuitiva e natural, sem a necessidade de um dispositivo externo.

Identify applicable funding agency here. If none, delete this.

B. Objetivos

O objetivo deste projeto é criar um programa em Python capaz de mapear a mão em tempo real e inferir qual gesto está sendo realizado pelo usuário, com o propósito de controlar o computador através de um mouse virtual. O programa utilizará as bibliotecas OpenCV e MediaPipe Hands para realizar o mapeamento e classificação dos gestos.

Além disso, pretende-se realizar testes para avaliar a eficácia e a precisão do programa em reconhecer os gestos realizados pelo usuário. Também serão realizados testes para verificar a capacidade do programa em controlar o computador através do mouse virtual em diferentes contextos de uso.

C. Contribuições

Este projeto tem como contribuições principais a possibilidade de oferecer uma interação mais intuitiva e natural com o computador, especialmente para pessoas que possuem dificuldades em utilizar o mouse convencional. Além disso, o programa pode ser útil em situações em que o mouse físico não está disponível, como em apresentações ou em casos de danos ao equipamento.

O desenvolvimento do programa também pode contribuir para a evolução da área de reconhecimento de gestos, uma vez que serão utilizadas bibliotecas de código aberto para realizar o mapeamento e classificação dos gestos. Isso pode possibilitar a criação de outros programas que utilizam essa mesma tecnologia para outras aplicações na área de interação homem-máquina.

II. Revisão de Literatura

O reconhecimento de gestos é uma área de pesquisa interdisciplinar que envolve processamento de imagens, aprendizado de máquina e inteligência artificial. Na literatura, os gestos são definidos como movimentos corporais expressivos que podem ser interpretados como sinais de intenção de um indivíduo [1]. O reconhecimento de gestos da mão em particular tem recebido uma atenção significativa nos últimos anos, com a popularidade crescente de interfaces de usuário baseadas em gestos, tais como a Kinect da Microsoft e a tecnologia Leap Motion [2].

Uma técnica comum de reconhecimento de gestos é o uso de câmeras e sensores para capturar o movimento do corpo

humano em tempo real [3]. No projeto proposto, a biblioteca OpenCV será utilizada para processar imagens capturadas por uma câmera e a biblioteca MediaPipe Hands será utilizada para mapear e classificar gestos das mãos. A biblioteca OpenCV é amplamente utilizada na comunidade de visão computacional devido à sua eficiência e facilidade de uso. Ela fornece recursos para capturar imagens de câmeras, manipulação de imagens e processamento de vídeo [4]. A biblioteca MediaPipe Hands, por sua vez, é uma biblioteca de aprendizado de máquina de visão computacional que pode detectar e rastrear as mãos em tempo real [5].

Um dos principais desafios da área de reconhecimento de gestos é a variabilidade natural dos gestos humanos. Gestos podem variar em forma, tamanho, velocidade e orientação. Além disso, as condições de iluminação e o posicionamento da câmera podem afetar a qualidade das imagens capturadas. A detecção de gestos também pode ser dificultada por obstáculos na cena ou por outras partes do corpo que possam ser confundidas com as mãos.

III. Desenvolvimento

A tarefa de obter ações a partir de gestos reconhecidos por câmera é complexa devido à quantidade de informações contidas em cada imagem. Por exemplo, uma imagem de 1280 x 720 pixels com codificação RGB possui uma grande quantidade de valores a serem processados. Para superar esse desafio e permitir a criação de trabalhos com gestos de forma precisa, a Google desenvolveu o pacote MediaPipe Hands.

De acordo com a documentação oficial do MediaPipe Hands [5], essa biblioteca utiliza uma combinação de algoritmos de aprendizado de máquina e visão por computador para detectar e estimar a posição das mãos em tempo real. Especificamente, ela se vale de redes neurais convolucionais (CNNs) para realizar a detecção e o rastreamento das mãos. As CNNs são um tipo de algoritmo de aprendizado de máquina que tem a capacidade de reconhecer padrões complexos em dados visuais, possibilitando a identificação precisa das mãos em imagens e vídeos.

Deste modo a utilização do MediaPipe Hands juntamente com a biblioteca OpenCV foi a escolha feita para a realização do projeto em questão. Basicamente, a solução proposta pela Google transforma a imagem de entrada em um vetor de vetores, onde cada vetor representa uma marcação específica das mãos. São fornecidas 21 marcações que indicam a posição das mãos no espaço bidimensional, com coordenadas x e y como podemos ver na figura 1.

A. Obtenção dos Dados

Graças às tecnologias escolhidas, temos uma base solida para iniciar, uma vez que a parte mais complexa da classificação de gestos já foi implementada. No entanto, para avançar na tarefa de controlar ações em um computador por meio de gestos, é necessário criar um banco de dados inicial contendo os gestos a serem treinados.

Com esse objetivo em mente, foi desenvolvida uma função que utiliza a câmera para capturar em tempo real 1000 imagens

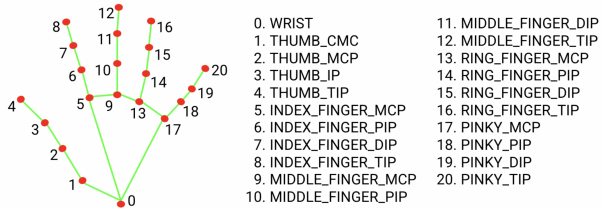

Fig. 1: Marcações MediaPipe Hands

Fonte: https://developers.google.com/mediapipe

consecutivas, salvando-as em arquivos separados para cada gesto desejado. Posteriormente, essas imagens são processadas em outro arquivo Python, no qual cada imagem é submetida ao MediaPipe Hands para obter as marcações das mãos, conforme ilustrado na figura 2.

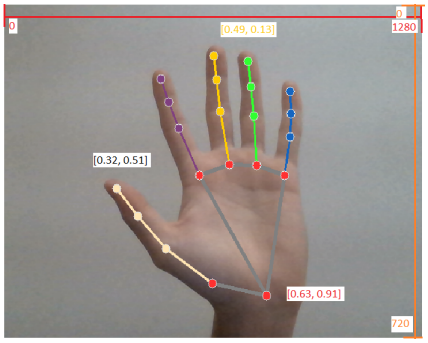

Fig. 2: Exemplo Marcações MediaPipe Hands

Fonte: Autoria Própria

B. Normalização dos Dados

Ao analisarmos a figura 2, fica evidente que o retorno do MediaPipe Hands consiste em valores de coordenadas x e y que representam a posição da mão em relação à câmera. Por exemplo, o ponto vermelho está aproximadamente a 63% da largura da câmera e 91% da altura da câmera. Esses valores variam de *zero* a *um* e a distância entre cada ponto varia dependendo da posição da mão, da distância da câmera ou da rotação da mão. Essa variabilidade torna a inferência do gesto da mão mais complexa e menos precisa.

Para solucionar esse problema, é necessário normalizar esses valores. Para isso, escolhemos a marcação *zero* da mão como ponto de referência, e usamos essa marcação como base para calcular as distâncias relativas aos demais pontos. Em seguida, dividimos essas distâncias pela maior distância entre os pontos, como ilustrado na figura 3. Portanto agora temos todos os pontos normalizados entre valores de menos um a um positivo, salvos em um dataset que será usado para treinar nosso modelo de predição.

C. Treinamento dos Modelos

Após a criação do dataset, o próximo passo é treinar um modelo de inteligência artificial para realizar a classificação. Para esse propósito, foram escolhidos dois algoritmos, Random Forest Classifier e o SVC, ambos são do scikit-learn,

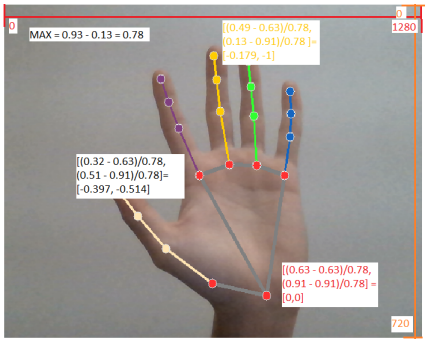

Fig. 3: Normalização

Fonte: Autoria Própria

que é uma biblioteca popular para aprendizado de máquina em Python.

O Random Forest é um algoritmo de aprendizado supervisionado que utiliza uma coleção de árvores de decisão para realizar a classificação. Cada árvore na floresta é construída a partir de uma amostra aleatória do conjunto de treinamento e a classificação final é determinada pela média das classificações individuais de cada árvore [6].

Já o SVC é baseado na técnica de máquinas de vetores de suporte e é usado para classificar amostras em diferentes classes com base em um conjunto de características (features). O SVC funciona construindo um hiperplano de separação entre as diferentes classes, onde um hiperplano é uma generalização de um plano em um espaço de dimensão superior.

Os dados foram divididos em conjuntos de treinamento e teste, onde 20% dos dados foram alocados para o conjunto de teste e o restante para o conjunto de treinamento. O modelo foi treinado para classificar dez classes distintas. Nas figuras 4 e 5 estão as matriz de confusão obtidas durante a avaliação dos modelos Random Forest Classifier e SVC respectivamente.

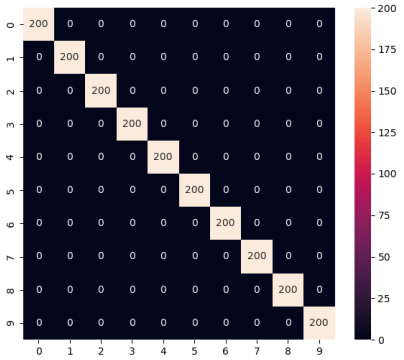

Fig. 4: Matriz de Confusão Random Florest

Fonte: Autoria Própria

Como pode ser observado, ambos os modelos apresentaram uma precisão de 100% na classificação dos dez gestos. Portanto, ambos os modelos são excelentes opções para o projeto

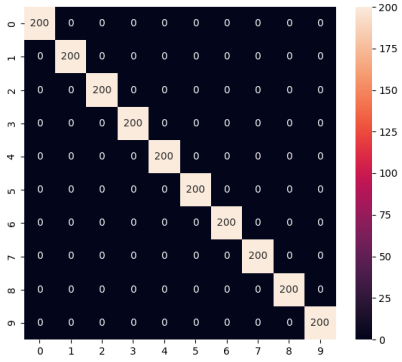

Fig. 5: Matriz de Confusão SVC

Fonte: Autoria Própria

proposto. Como precisamos escolher um modelo, foi escolhido o Random Florest Classifier para prosseguir com o projeto.

D. Controlando o Mouse

Para controlar o computador foram escolhidos 10 gestos que são mostrados na figura 6, onde eles representam:

TABLE I: Tabela de Gestos

Número	Gesto
1	Mouse
2	Cima
3	Baixo
4	Esquerdo
5	Direito
6	Fechado
7	Pinça
8	Aberto
9	Copiar
10	Colar

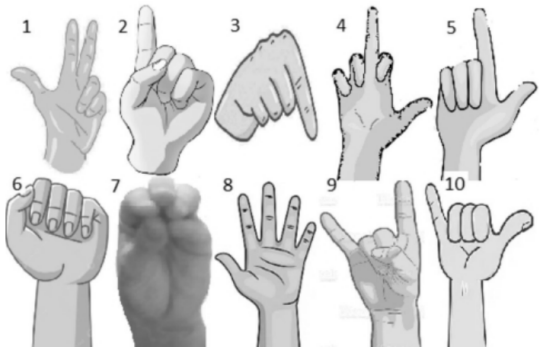

Fig. 6: Gestos usados

Fonte: Autoria Própria

Após o treinamento e validação do modelo, o sistema adquire a capacidade de reconhecer os dez gestos necessários. O próximo passo consiste na implementação da interação homem-máquina quando um gesto é reconhecido. Para isso,

foi utilizado o pacote PyAutoGUI, uma biblioteca que oferece recursos para controlar o cursor do mouse, executar scroll, usar hotkeys, entre outras funcionalidades.

Basicamente o programa compara o estado atual com o anterior para saber se um comando deve ser realizado.Na tabela 2 é possível observar todos as combinações de gestos e seus respectivos comandos resultantes.

TABLE II: Tabela de Gesto e Comando

Gesto Anterior	Gesto Atual	Comando
Mouse	Direito	Clique botão Direito
Mouse	Esquerdo	Clique botão Esquerdo
Aberto	Pinça	Zoom Out
Pinça	Aberto	Zoom In
-	Baixo	Rolagem para baixo
-	Cima	Rolagem para cima
-	Copiar	Hotkey Ctrl + C
-	Colar	Hotkey Ctrl + V
-	Fechado	Cursor Estático

A movimentação do cursor é feita através da variação da posição absoluta da mão em relação a câmera, pegando a marcação número um da mão. Muitas vezes ao navegar pelo computador, a mão pode acabar saindo da câmera. Por esse motivo, existe o gesto *fechado*, que ao ser reconhecido, deixa o cursor estático e o usuário pode voltar a sua mão para frente da câmera.

IV. Resultados e Conclusões

Durante o projeto, a seleção dos gestos para cada estado demonstrou resultados satisfatórios, pois os gestos escolhidos eram distintos entre si. Isso possibilitou inferências com uma precisão quase perfeita, independentemente do modelo utilizado, seja o Random Forest Classifier ou o SVC. A alta precisão das inferências é atribuída à escolha criteriosa dos gestos, que foram projetados com características bem diferenciadas. Essa clara diferenciação entre os gestos facilitou o processo de treinamento e resultou na correta classificação dos estados.

Apesar da classificação dos gestos em tempo real apresentar uma precisão quase perfeita, a implementação do reconhecimento de gestos por vídeo em conjunto com o uso do PyAutoGUI pode ser uma tarefa complexa e desafiadora. No computador em que o programa foi desenvolvido, foi observada uma taxa média de atualização de 15 FPS, mesmo com a utilização de um processador Core i5 de 11ª geração.

Essa limitação de desempenho pode impactar a experiência do usuário e restringir a aplicabilidade prática do sistema. É importante considerar otimizações e alternativas para melhorar a taxa de atualização e garantir uma resposta mais ágil e fluída do reconhecimento de gestos por vídeo.

Esses desafios ressaltam a necessidade contínua de aprimoramento e ajustes no sistema, visando alcançar uma execução mais eficiente e satisfatória, especialmente em termos de taxa de atualização, para proporcionar uma experiência de uso mais suave e responsiva aos usuários.

V. Conclusão

Em resumo, este trabalho abordou o desenvolvimento de um programa que utiliza reconhecimento de gestos por meio de câmera para controlar o mouse de um computador. A solução implementada foi baseada na biblioteca MediaPipe Hands da Google, combinada com a biblioteca OpenCV, para detectar e estimar a posição das mãos em tempo real.

Foi realizada a obtenção de dados, capturando imagens da câmera e processando-as para obter as marcações das mãos. Em seguida, os dados foram normalizados e utilizados para treinar dois modelos de classificação: Random Forest Classifier e SVC. Ambos os modelos apresentaram alta precisão na classificação dos gestos, alcançando 100% de acurácia.

Para controlar o mouse, foram definidos 10 gestos representando diferentes comandos. O programa utiliza o estado atual e o estado anterior dos gestos para determinar as ações a serem realizadas. O pacote PyAutoGUI foi empregado para executar as operações de controle do mouse, como cliques, rolagem e atalhos de teclado.

No entanto, foi observada uma limitação de desempenho em termos de taxa de atualização, com uma média de 15 FPS no computador utilizado para o desenvolvimento. Essa limitação pode afetar a experiência do usuário, tornando necessário explorar melhorias e otimizações para alcançar uma taxa de atualização mais alta e proporcionar uma resposta mais ágil e fluída no reconhecimento de gestos.

Apesar dos desafios enfrentados, este trabalho representa um passo importante na utilização de reconhecimento de gestos para o controle do mouse. A precisão alcançada na classificação dos gestos e a implementação funcional do controle do mouse demonstram o potencial dessa abordagem como uma forma intuitiva e interativa de interagir com computadores. Com refinamentos adicionais e considerações de desempenho, esse sistema pode encontrar aplicação em áreas como jogos, acessibilidade e interfaces de usuário inovadoras.

References

[1] Pavlovic, V. I., Sharma, R., & Huang, T. S. (1997). Visual Interpretation of Hand Gestures for Human-Computer Interaction: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7), 677-695. doi: 10.1109/34.598226.

[2] Oikonomidis, I., Kyriazis, N., & Argyros, A. A. (2011). Efficient model-based 3D tracking of hand articulations using Kinect. In *Proceedings of the British Machine Vision Conference* (pp. 1-11). doi: 10.5244/C.25.101

[3] Devineau, G., Xi, W., Moutarde, F., & Yang, J. (2018). Deep Learning for Hand Gesture Recognition on Skeletal Data. In *Proceedings of the 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)* (pp. 119-126). doi: 10.1109/FG.2018.00025.

[4] Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 25, 120-125.

[5] Google. MediaPipe Hands. Disponível em: *https://developers.google.com/mediapipe/solutions/vision/handlandmarker*. Acesso em: 22 maio 2023.

[6] Scikit-learn. RandomForestClassifier. Disponível em: *https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html*. Acesso em: 22 maio 2023.