TRANSFERÊNCIA DE ARQUIVOS MULTIPLATAFORMA POR MEIO DE RECONHECIMENTO DE GESTOS

Gustavo Vaz Teixeira[1], Alessandra Martins Coelho[2]

[1]Instituto Federal do Sudeste de Minas Gerais - Campus Rio Pomba, Rio Pomba, Brasil (guteixeira2001@gmail.com)
[2]Instituto Federal do Sudeste de Minas Gerais - Campus Rio Pomba, Rio Pomba, Brasil

Resumo: A transferência de arquivos entre dispositivos heterogêneos tem se tornado cada vez mais comum em diferentes contextos e plataformas, exigindo, muitas vezes um certo nível de conhecimento técnico e carga cognitiva, principalmente para usuários sem familiaridade com computadores ou com dificuldades motoras. A integração entre sistemas distribuídos e visão computacional pode permitir soluções inovadoras voltadas à comunicação e transferência de arquivos entre diferentes plataformas, de maneira inteligente e mais próxima da interação humana. Com isso em mente, o presente trabalho possui como principal objetivo desenvolver e avaliar uma abordagem distribuída baseada em reconhecimento de gestos para transferência de arquivos entre dispositivos heterogêneos. Os procedimentos metodológicos adotados no desenvolvimento do trabalho foram organizados em cinco etapas principais, sendo essas a pesquisa bibliográfica para escolha de tecnologias, o desenvolvimento das vertentes do sistema, implementação do modelo de reconhecimento de gestos, a realização de testes e análise de resultados. Nos experimentos realizados, foi realizada uma comparação com os métodos tradicionais Google Drive e Bluetooth, observou-se que a presente proposta apresentou um tempo menor de transferência, um número inferior de etapas e uma estimativa qualitativa de baixa carga cognitiva, uma vez que boa parte da interação com o sistema se baseia no reconhecimento de gestos. Concluímos que a combinação entre reconhecimento de gestos e sistemas distribuídos se mostrou eficaz na transferência de arquivos em tempo real.

Palavras-chave: Gestos; Arquivos; Transferência; Visão Computacional; Sistemas Distribuídos.

INTRODUÇÃO

A transferência de arquivos entre diferentes dispositivos tem se tornado cada vez mais comum em diferentes contextos, como a academia, o mercado de trabalho e se estendendo até o dia a dia de usuários comuns. Essa transferência vai além de sistemas e dispositivos homogêneos (Simeone et al., 2013), muitas vezes exigindo conhecimento técnico prévio em diferentes sistemas operacionais (Android, Windows, etc.), diferentes interfaces e meios de comunicação, como por exemplo, o Google Drive e o Bluetooth.

Esse grande número de interfaces impacta diretamente no número de ações necessárias, aumentando a carga cognitiva para realizar uma simples ação de transferência de arquivos, principalmente para usuários sem conhecimento técnico ou com dificuldades motoras. Moreno et al. (2024), argumentam que os esforços realizados para deixar a internet acessível são insuficientes, principalmente para usuários com dificuldades cognitivas, enquanto Hoehl (2016) afirma que usuários com dificuldades cognitivas não se sentem confortáveis em realizar compras on-line, uma tarefa comum do século moderno.

Avanços em sistemas distribuídos têm desempenhado um papel fundamental na transferência e sincronização de dados entre dispositivos heterogêneos. Segundo Tanenbaum e Van Steen (2007), esses sistemas são projetados para integrar computadores e redes heterogêneas, oferecendo uma visão unificada dos recursos disponíveis aos usuários por meio de camadas intermediárias de software.

Em paralelo, os algoritmos de visão computacional também possibilitaram melhorias na detecção de objetos e reconhecimento de gestos, com possibilidade de aplicação até mesmo no reconhecimento de gestos ofensivos (Rodrigues, Narcizo e Shimanuki, 2024). A integração entre essas tecnologias permite soluções inovadoras voltadas à comunicação e compartilhamento de arquivos entre

plataformas diferentes, de maneira inteligente e mais próxima da interação humana.

Diante dos pontos levantados, este trabalho propõe o desenvolvimento de uma abordagem distribuída para transferência de imagens a partir do reconhecimento de gestos, especificamente o gesto "pegar", uma mão fechada e o gesto "soltar", delimitado por pelo menos dois dedos separados. A principal proposta é permitir que usuários realizem o envio de arquivos entre diferentes ambientes computacionais por meio de gestos em tempo real, proporcionando uma experiência mais fluida para transferência de imagens e desenvolvimento de interfaces naturais a partir do uso de câmera.

A relevância deste estudo está na exploração de diferentes soluções disponíveis na Ciência da Computação, como sistemas distribuídos e visão computacional para novas maneiras de interação digital, contribuindo para o avanço de soluções tecnológicas, acessibilidade e experiência do usuário. O impacto na acessibilidade motora se apoia na drástica diminuição do número de interações com o computador necessárias para transferência de imagens.

O objetivo geral do presente trabalho é desenvolver e avaliar uma abordagem distribuída baseada em reconhecimento de gestos para transferência de arquivos entre dispositivos heterogêneos. Para os objetivos específicos, destacam-se (i) implementar mecanismos de visão computacional para detecção de gestos; (ii) estruturar a comunicação distribuída entre dispositivos heterogêneos a partir da arquitetura cliente-servidor; (iii) avaliar o número de interações com o computador, tempo de transferência e carga cognitiva necessárias para transferência de arquivos; (iv) realizar uma comparação com os principais métodos de transferência de arquivos, Google Drive e Bluetooth.

Para atingir tais objetivos, uma abordagem experimental será utilizada, envolvendo etapas que vão desde a pesquisa bibliográfica e desenvolvimento de protótipos e testes práticos em ambiente controlado.

Espera-se que os resultados deste trabalho contribuam para a discussão sobre interfaces naturais de interação por meio da integração de diferentes áreas da Ciência da Computação, demonstrando a viabilidade da utilização do reconhecimento de gestos na transferência de arquivos entre dispositivos heterogêneos.

O trabalho está organizado em seis seções, sendo a segunda seção focada nos objetivos. Na terceira seção, são organizados e separados trabalhos disponíveis na literatura, com foco em sistemas distribuídos, transferência de arquivos e utilização de JavaScript em diferentes contextos. Na quarta seção, está descrita a metodologia do trabalho. Partindo para a quinta seção, são discutidos os resultados obtidos pelo trabalho e também é realizada uma comparação com os métodos tradicionais para transferência de arquivos. Por fim, na última seção é realizada a conclusão.

MATERIAL E MÉTODOS

A metodologia desenvolvida neste trabalho pode ser classificada principalmente como uma pesquisa exploratória, uma vez que foram coletados e analisados trabalhos relacionados à transferência de arquivos por meio dos gestos, além da necessidade de implementação de uma lógica própria para o desenvolvimento do modelo de gestos e também pelos testes do modelo de reconhecimento gestual em diferentes dispositivos. O trabalho também possui aspectos de uma pesquisa descritiva, visto que a análise dos resultados foi baseada em métricas como tempo de resposta, quantidade de interações necessárias, precisão no reconhecimento de gestos e análise de carga cognitiva.

Os procedimentos metodológicos adotados no desenvolvimento deste trabalho foram organizados em cinco etapas principais, as quais estão ilustradas pelo fluxograma da Figura 7, sendo estas: (i) Pesquisa bibliográfica e escolha de tecnologias; (ii) Implementação das duas vertentes do sistema; (iii) Implementação do modelo de reconhecimento de gestos; (iv) realização de testes; (v) análise dos resultados. Essas etapas serão discutidas com mais detalhes nos próximos parágrafos.

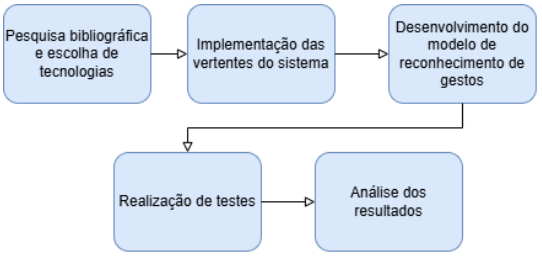

Figura 1. Fluxograma do Projeto

A primeira etapa do trabalho concentrou-se na pesquisa bibliográfica e levantamento de tecnologias de ponta, na qual foram coletados e analisados trabalhos relacionados ao uso de tecnologias web para visão computacional, com ênfase em reconhecimento de gestos.

Foram levantados quatro trabalhos principais: Peer-to-Peer File Streaming Using WebSockets

Protocol, Reconhecimento de Gestos de Mãos Ofensivos em Contextos Multiculturais Utilizando LLM, NoTouch.js: A JavaScript Library for Touch-Free Web Browsing e A Cross-Device Drag-and-Drop Technique, os quais se revelaram extremamente úteis para escolha das tecnologias utilizadas neste trabalho. Ressalta-se que os trabalhos listados na revisão bibliográfica não possuíam o mesmo objetivo que o presente trabalho, entretanto, foi observada a eficácia do uso de WebSockets e MediaPipe, principais tecnologias utilizadas para realizar a transferência de arquivos pelo reconhecimento de gestos.

Partindo para a segunda etapa, o foco se concentrou no desenvolvimento distribuído do projeto, o qual obteve duas vertentes, a versão web desenvolvida em React e Next.js e a versão mobile desenvolvida em React Native. A escolha dessas tecnologias se deu pelo uso e suporte da mesma linguagem de programação (TypeScript) e pelo suporte à comunicação entre dispositivos por Socket.IO, que permite comunicação em tempo real e bidirecional entre clientes e servidor, utilizando WebSockets.

Para o desenvolvimento da vertente Web, foi escolhido o React. A escolha do React em detrimento de outros frameworks como Vue.js e Angular foi baseada principalmente na reutilização de conhecimento e código. Essa escolha permitiu padronizar o desenvolvimento entre diferentes plataformas, uma vez que a mesma base conceitual foi aplicada na versão web com React e para a versão mobile por meio do React Native.

A escolha do React Native para o desenvolvimento da versão mobile no lugar de Ionic ou outros frameworks de desenvolvimento multiplataforma híbrido se dá pelo uso de mesma linguagem de programação e também pela familiaridade com a versão Web, que também utilizaram o React como base.

Além das duas vertentes baseadas no lado do cliente, o sistema desenvolvido também conta com um servidor responsável pela comunicação entre os clientes, desenvolvido em Express. A arquitetura segue o modelo cliente-servidor, utilizando WebSockets por meio da biblioteca Socket.IO, permitindo a comunicação em tempo real e de forma bidirecional.

No fluxograma da Figura 8, está ilustrada como os clientes e o servidor se comunicam e como é realizada a transferência de arquivos. Quando um usuário realiza o gesto de envio (pegar), o arquivo é enviado para o servidor e o servidor notifica os demais clientes conectados que uma transferência foi iniciada. Caso algum cliente aceite a operação, o arquivo é então transferido para outro cliente, ficando temporariamente na memória do receptor e posteriormente removido do servidor após a conclusão da transferência. Essa abordagem é adotada, pois o servidor não tem como objetivo atuar como sistema de armazenamento permanente.

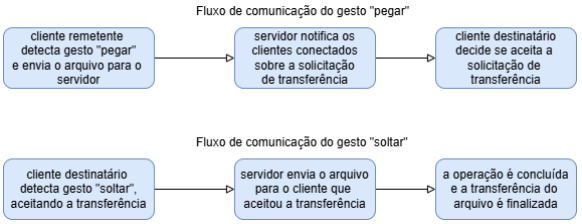

Figura 2: Fluxograma da transferência de arquivos.

A terceira etapa concentrou-se na implementação do modelo de reconhecimento de gestos “pegar” e “soltar”, na qual foi utilizada a biblioteca MediaPipe, do Tensorflow.js. Essa implementação específica não foi feita utilizando apenas a classificação direta fornecida pela MediaPipe, sendo necessário implementar uma lógica própria para realizar a classificação, baseada na análise da proximidade entre pontos de referência da mão.

Inicialmente, os gestos “pegar” e “soltar” eram definidos simplesmente por abrir e fechar a mão. Notou-se, entretanto, que essa definição não seria inclusiva, uma vez que muitas pessoas não possuem coordenação motora ou os cinco dedos das mãos, para realizar os gestos propostos. Para resolver esse problema, foi proposta uma implementação que utilizava distância euclidiana entre as extremidades dos dedos, os quais foram definidos usando os 21 pontos da mão pela MediaPipe.

Com os 21 pontos da mão definidos, foi implementada uma lista de pares que representava o polegar com os outros quatro dedos e os outros dedos entre si. Essa lista pode ser encontrada na Tabela 1, a qual descreve melhor cada uma combinação e sua representação.

Tabela 1. Pares de pontos utilizados para detecção de proximidade entre dedos.

Primeiro valor	Segundo valor	Gesto
4	8	polegar e indicador
4	12	polegar e médio
4	16	polegar e anelar
4	20	polegar e mínimo
8	12	indicador e médio
12	16	médio e anelar
16	20	anelar e mínimo
8	20	indicador e mínimo

Uma vez que as combinações foram definidas, foi possível calcular a distância euclidiana entre os dedos, utilizando um threshold. O threshold foi definido com base na distância entre os pontos da imagem. A definição do gesto "pegar" foi interpretada como pelo menos três pares de pontos com a distância inferior ao limiar estabelecido, ou seja, cujo threshold era menor que 45, os quais podem ser qualquer combinação de dedos. E o gesto "soltar" foi definido pela falta de dedos próximos, ou seja, a distância euclidiana maior que o threshold, representando uma mão "aberta". Ressalta-se que o valor para o threshold foi definido empiricamente, e pode variar conforme a resolução da imagem e a distância da mão em relação a câmera.

Devido ao seu suporte a navegadores, a aplicação dessa biblioteca na versão web foi de implementação simplificada, sendo necessário apenas baixar as dependências e implementar o reconhecimento de gestos com base nas regras definidas previamente.

Apesar de sua familiaridade com as outras versões, a vertente mobile apresentou uma maior complexidade de implementação. O React Native não foi projetado para executar algoritmos de visão computacional em tempo real, enquanto o MediaPipe foi originalmente desenvolvido para execução em ambiente Web. Dessa forma, tornou-se necessário realizar comunicação com recursos nativos do smartphone utilizando bridges para integração com código Android nativo.

A bridge desenvolvida utilizou o expo-modules em combinação com a solução MediaPipe Tasks Gesture Recognizer, a qual provê código Android nativo. A partir dessa solução, tornou-se possível a utilização do MediaPipe pelo React Native. Na Figura 9, está ilustrado um diagrama da arquitetura da bridge. Na primeira etapa, desenvolve-se a UI, captura o gesto e realiza uma chamada para a bridge. Na segunda etapa, a bridge intermedia a comunicação entre a interface JavaScript e o código Kotlin e também serializa os dados. Entrando no Android nativo, a câmera é acessada, captura os frames e faz uma chamada para execução do MediaPipe. Por fim, na quarta etapa o modelo de visão computacional "gesture_recognizer.task" é carregado, realiza a inferência em tempo real e então retorna com os resultados para o React Native.

Durante a execução, observou-se que o modelo "gesture_recognizer.task" apresentava bom desempenho na identificação dos elementos previamente suportados pela solução. Entretanto, para os gestos de "mão aberta" e "mão fechada", utilizados neste trabalho, a taxa de acerto dificilmente ultrapassava 70%, exigindo adaptações complementares na lógica de reconhecimento. Apesar dessa limitação, o desempenho obtido foi suficiente para o funcionamento adequado do sistema.

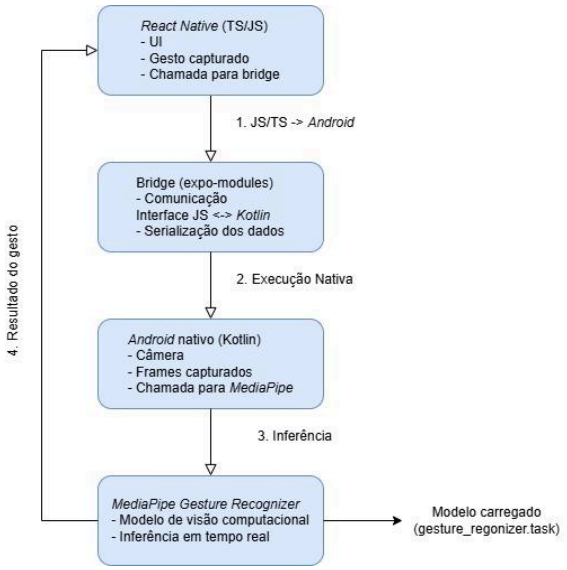

Figura 3: Arquitetura da *bridge* desenvolvida

A quarta etapa consistiu na realização de testes de usabilidade e desempenho das três vertentes desenvolvidas. Os aspectos avaliados foram o tempo de resposta, quantidade de interações necessárias para a transferência de arquivos, desde a seleção do arquivo, precisão no reconhecimento de gestos e também a estabilidade da aplicação durante o tempo de execução.

Os testes foram executados em diferentes dispositivos e também em diferentes condições de ambiente, como a iluminação e distância da tela, com o objetivo de identificar possíveis limitações e falhas no sistema.

Os dispositivos utilizados foram um Notebook Acer Nitro 5, para avaliação das versões Web e Desktop, enquanto a versão mobile foi testada em um smartphone Xiaomi Redmi Note 14 Pro 5g. Foram consideradas diferentes condições para testagem da iluminação, incluindo apenas a iluminação dos dispositivos e o uso de iluminação artificial adicional, por meio de um iluminador.

Por fim, a última etapa concentrou-se na análise dos dados obtidos durante os testes, possuindo como principal objetivo avaliar o desempenho do sistema. Foram realizadas comparações com métodos tradicionais de transferência de arquivos, como Bluetooth e Google Drive, comparando métricas como tempo de transferência e número de interações necessárias. Além disso, essa etapa também buscou identificar padrões de falhas, limitações do sistema e os impactos do uso de JavaScript/TypeScript em cenários de processamento em tempo real,

comunicação distribuída e visão computacional. Esses resultados foram listados na Seção Resultados e Discussão.

A utilização de inteligência artificial no presente trabalho se concentrou em diferentes modelos, empregados com diferentes finalidades. O principal modelo utilizado foi o Github Copilot, integrado ao Visual Studio Code. Esse modelo foi empregado principalmente para sugestões de códigos, autocompletar e auxílio na tipagem, uma vez que o desenvolvimento em TypeScript exige maior rigor na definição de tipos e estruturas de dados. Entretanto, algumas vezes o GitHub Copilot acabava alucinando e sugerindo coisas que não faziam sentido, necessitando uma análise técnica crítica do código sugerido pelo modelo.

O Stitch, desenvolvido pela Google, foi utilizado para a geração inicial de protótipos de interface, servindo como fonte de inspiração durante o processo de design. Porém, as interfaces geradas apresentavam uma grande quantidade de elementos visuais desnecessários, como botões, menus, componentes de navegação, entre outros. Na Figura 10, está ilustrado um exemplo de interface gerado pelo Stitch, com intuito apenas de listar os arquivos disponíveis. Observa-se que a interface possui aproximadamente 28 botões.

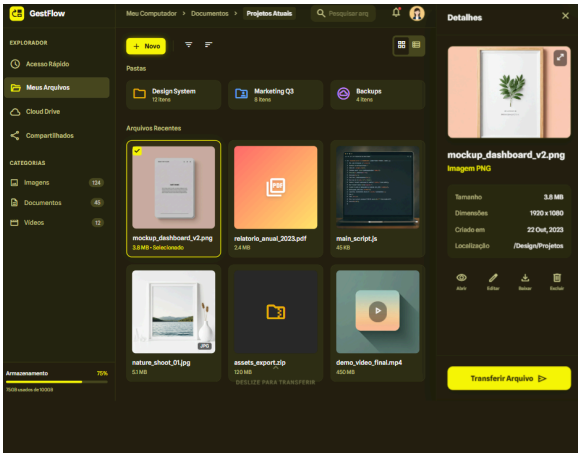

Figura 4. Exemplo de interface ”arquivos” gerada pelo Stitch.

Considerando que a proposta busca reduzir a dependência de interfaces tradicionais, como botões, por meio da interação baseada em gestos, tornou-se necessário realizar adaptações significativas e simplificações na interface gerada pela ferramenta. Na Figura 11, está a mesma interface da Figura 10, porém, com uma diminuição significativa de elementos visuais sugeridos pelo Stitch.

Figura 5. Exemplo de interface ”arquivos” gerada pelo Stitch.

O modelo Claude também foi utilizado como ferramenta de apoio durante o desenvolvimento, principalmente na investigação e correção de problemas técnicos mais complexos, os quais apareceram principalmente na vertente mobile. A arquitetura do React Native com Expo e diferentes componentes apresentaram desafios que, em diversos momentos, dificultaram a obtenção de sugestões eficazes por meio do GitHub Copilot.

Entretanto, determinadas funcionalidades exigiram consulta direta à documentação oficial das tecnologias utilizadas. Um exemplo foi a implementação da bridge desenvolvida para o projeto, cuja complexidade ultrapassou a capacidade dos modelos de inteligência artificial empregados. O mesmo ocorreu na implementação do modelo da MediaPipe em código nativo para dispositivos móveis, na qual as ferramentas de inteligência artificial contribuíram principalmente para ajustes de sintaxe, enquanto a solução de tais problemas necessitou de consultas da documentação técnica das ferramentas.

Por fim, o ChatGPT foi utilizado como ferramenta de apoio à escrita e documentação do projeto. A principal utilização do modelo se concentrou na revisão textual e no aprimoramento de textos, por meio de prompts como “Como posso melhorar o texto abaixo?” e “Há informações o bastante para o entendimento do parágrafo abaixo?”. Dessa forma, o modelo contribuiu para a organização das ideias e para a melhora da clareza dos textos produzidos.

No entanto, foram observadas algumas limitações durante sua utilização. Como foi utilizada a versão gratuita, o modelo apresentava inconsistências em relação ao contexto previamente estabelecido ou sugeria informações, funcionalidades e propostas que não faziam parte do presente trabalho. Por esse motivo, todas as sugestões fornecidas foram submetidas a uma análise crítica, sendo utilizadas apenas quando compatíveis com os requisitos, objetivos e implementação efetivamente realizados no projeto.

Na Tabela 2, foram listados os modelos de inteligência artificial utilizados e as limitações apresentadas por cada um deles.

Tabela 2. Modelo de IA utilizado e limitação apresentada.

Modelo de IA	Limitação apresentada
Github Copilot	Apresentou inconsistências em códigos mais complexos.
Stitch	Geração desnecessária de muitos elementos visuais, fugindo do que foi solicitado
Claude	Apresentou limitações em componentes complexos
ChatGPT	Inconsistência e perda de contexto e sugestões fora do escopo do projeto

RESULTADOS E DISCUSSÃO

Um dos principais problemas levantados durante os testes foi a alta taxa de falsos positivos apresentados pelo modelo da MediaPipe da web. Uma das soluções avaliadas para a resolução desse problema foi criar um modelo próprio utilizando o TeachableMachine, da Google. Foram utilizadas 100 imagens para treinamento do gesto “pegar” e outras 100 imagens para o gesto soltar. Entretanto, os modelos da Teachable Machine precisam ser treinados com muito mais cautela, uma vez que eles reconhecem tudo ao redor, ou seja, foram ainda mais propensos a falsos positivos que o modelo da MediaPipe. Após uma melhora na iluminação, o modelo melhorou sua performance, sendo essa uma das limitações apresentadas pelo sistema proposto.

A utilização de JavaScript/TypeScript como principal linguagem de programação se mostrou eficiente para o desenvolvimento das vertentes web e mobile, porém, o JavaScript não se mostrou suficiente para atender integralmente aos requisitos da proposta, sendo necessário o apoio de tecnologias externas, como pacotes do npm e, no caso mobile, bridges nativas. Outro problema associado ao uso dessa linguagem é o tamanho dos projetos, decorrente principalmente da grande dependência de muitos pacotes baixos via npm.

A comparação com métodos tradicionais de transferência de arquivos foi realizada considerando métricas como quantidade de interações necessárias, tempo de transferência e carga cognitiva associada ao processo de interação. Para cada ferramenta, foram realizados 10 testes em ambiente controlado pela mesma pessoa, utilizando o mesmo arquivo e as mesmas condições experimentais. Os tempos apresentados correspondem à média aritmética obtida nos testes.

A média em segundos que o sistema levou para realizar a transferência web-mobile e mobile-web foi de aproximadamente 15 segundos. Um fato a destacar é que cerca de 30% a 40% desse tempo foi dedicado a carregar os modelos de reconhecimento de gestos na vertente web. As etapas necessárias para transferência de arquivos entre o Google Drive, Bluetooth e o presente trabalho estão descritas na Figura 12. Para uma comparação justa, todas as plataformas começaram a partir da seleção do arquivo e também, o mesmo arquivo foi utilizado em todas as ferramentas de transferência, uma imagem de tamanho aproximado a um megabyte. Para o presente trabalho, as etapas foram: (i) Seleção do arquivo a ser transferido; (ii) acionamento da funcionalidade de transferência; (iii) execução do gesto de “pegar” do arquivo; (iv) execução do gesto “soltar” direcionado ao dispositivo de destino.

O Google Drive utilizou as etapas de: (i) seleção do arquivo a ser transferido; (ii) acionamento da funcionalidade de upload para a nuvem; (iii) acesso ao serviço de armazenamento em nuvem no dispositivo de destino; (iv) localização do arquivo armazenado; (v) realização do download do arquivo no dispositivo de destino.

Por fim, para o Bluetooth, foram necessárias as seguintes interações: (i) seleção do arquivo a ser transferido; (ii) acionamento da funcionalidade de compartilhamento; (iii) ativação do Bluetooth no dispositivo de destino; (iv) seleção do dispositivo de destino entre as opções de compartilhamento disponíveis; (v) acesso ao modo de transferência Bluetooth no dispositivo receptor; (vi) acionamento da opção de recebimento de arquivos via Bluetooth; (vii) confirmação e aceitação da transferência no dispositivo de destino.

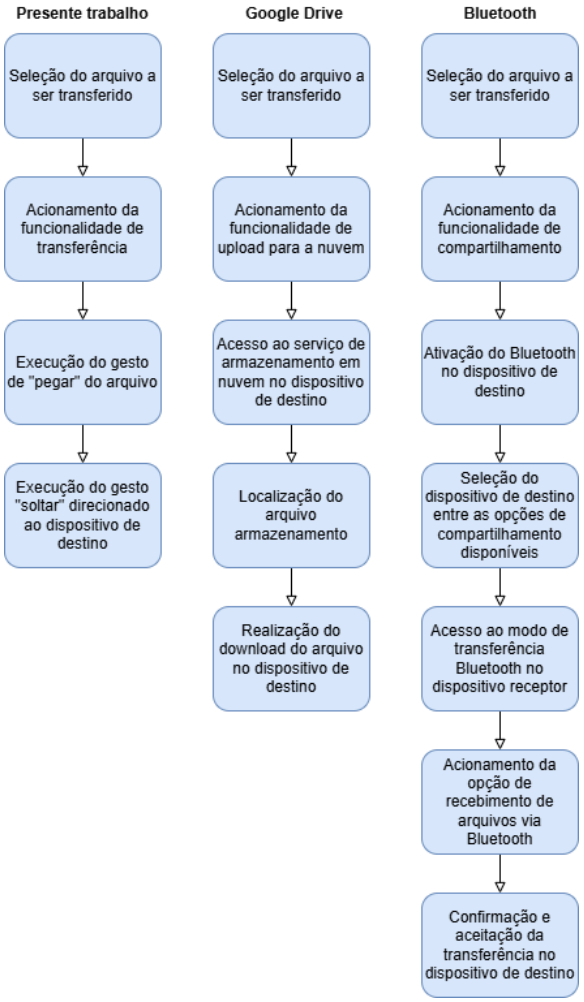

Figura 6. Fluxograma das etapas necessárias para transferência de arquivos entre diferentes dispositivos.

Além da comparação técnica, também foi realizada uma comparação do tempo de transferência, do número de interações necessárias com o computador e de uma estimativa qualitativa da carga cognitiva, que estão presentes na Tabela 3. Observou-se que o Bluetooth foi a ferramenta que apresentou maior estimativa qualitativa de carga cognitiva, exigindo conhecimento técnico prévio (ativar, parear, modos de envio/recebimento em diferentes sistemas operacionais), além do tempo de espera e transferência, apresentando tempo de transferência aproximadamente duas vezes maior que o sistema proposto. O Google Drive apresentou uma estimativa qualitativa de carga cognitiva moderada, sendo necessário realizar upload do arquivo, troca de dispositivos físicos e navegação para localizar o arquivo no outro dispositivo. Em relação ao tempo de transferência foi levemente superior com apenas 3 segundos a mais, porém é necessário conhecimento prévio da ferramenta e da interface do sistema. O presente trabalho apresentou o menor número de interações necessárias durante a transferência de arquivos, o que indica potencial para reduzir a carga cognitiva em relação aos métodos tradicionais.

Tabela 3. Comparação entre as ferramentas.

Ferramenta	Etapas	Estimativa qualitativa da carga cognitiva	Tempo (s)
Bluetooth	7	Elevada	40
Google Drive	5	Moderada a elevada	18
Presente trabalho	4	Baixa	15

Uma das principais contribuições do presente trabalho é explorar e ampliar novas vertentes para transferência de arquivos, visando diminuir parte das interações tradicionais com computadores pela substituição de menus, botões, compartilhamento, pareamento, entre outros através de gestos reconhecidos por visão computacional.

O trabalho também apresenta contribuições técnicas após comprovar a viabilidade de combinação entre diferentes tecnologias como aplicações web, aplicações móveis, sistemas distribuídos, reconhecimento de gestos e comunicação em tempo real entre dispositivos heterogêneos.

Além disso, o trabalho está fortemente inserido no contexto de interfaces espaciais, área que tem recebido atenção crescente na indústria e da academia por meio de tecnologias como realidade aumentada, interações sem toque e reconhecimento de gestos. Nesse contexto, a solução proposta contribui para a investigação de novas formas de interação entre usuários e dispositivos. A relevância dessa abordagem também pode ser observada em iniciativas do mercado e da indústria, como o sistema "*AI Air File Sharing*" da empresa Huawei, que aponta uma tendência em interações espaciais e gestuais para compartilhamento de dados.

CONCLUSÃO

A transferência de arquivos entre dispositivos heterogêneos necessita de conhecimento técnico prévio em diferentes interfaces e configurações, além de frequentemente exigir múltiplas interações com o computador e impor uma carga cognitiva média a elevada aos usuários, podendo representar uma barreira para determinados grupos de usuários, especialmente aqueles com menor familiaridade tecnológica ou com limitações motoras.

Diante desses fatos, os objetivos propostos neste trabalho foram alcançados, demonstrando que é possível realizar transferência de arquivos em ambientes multiplataforma utilizando o reconhecimento de gestos em tempo real, integrando tecnologias web, comunicação distribuída e visão computacional.

Entretanto, a proposta também apresentou algumas limitações durante a fase de testes. O modelo desenvolvido apresentou falsos positivos na vertente web, além de depender da iluminação. Na vertente mobile, o reconhecimento de gestos “mão aberta” e “mão fechada” atingiu uma acurácia entre 60% e 70%, resultado considerado suficiente para validação do protótipo, embora ainda exista espaço para aprimoramentos. O modelo disponibilizado pela MediaPipe também precisou ser adaptado para reduzir limitações relacionadas à coordenação motora dos usuários, diminuindo a dependência de movimentos como esticar completamente os dedos ou fechar totalmente a mão. Já na vertente mobile, foi necessário realizar integração nativa para executar o modelo de reconhecimento de gestos em um smartphone.

Em trabalhos futuros, espera-se a implementação de suporte a arquivos de maior tamanho e diferentes tipos de arquivos, tais como vídeos, planilhas, pdfs, entre outros, com apoio de chunks e gerenciamento de fila. Espera-se também a implementação do sistema em um ambiente profissional e com múltiplos usuários, visando limitar a dependência de sistemas de terceiros.

REFERÊNCIAS

Akcura, K. NoTouch.js: A JavaScript Library for Touch-Free Web Browsing. 2025. Advisor: Dr. Marta Kersten-Oertel.

Hoehl, J. A. Exploring Web simplification for people with cognitive disabilities. 2016. Tese (Doutorado) – University of Colorado at Boulder, Boulder, 2016. Disponível em: <https://www.proquest.com/openview/ee86a7d4b5b84a0bdedd9606801209ea/1?pq-origsite=gscholar&cbl=18750 />. Acesso em 13 jun 2026.

Moreno, L.; Petrie, H.; Martínez, P.; Alarcon, R. Designing user interfaces for content simplification aimed at people with cognitive impairments. Universal Access in the Information Society, v. 23, p. 99–117, 2024. DOI: https://doi.org/10.1007/s10209-023-00986-z. Disponível em: <https://link.springer.com/article/10.1007/s10209-023-00986-z/>. Acesso em 13 jun 2026

Rodrigues, G. G. L.; Narcizo, F. B.; Shimanuki, M. T. Reconhecimento de gestos de mãos ofensivos em contextos multiculturais utilizando LLM. In: Anais do XIV Seminário de Iniciação Científica e Pesquisa do Litoral Norte (SIC-LN). Caraguatatuba: [s.n.], 2024. Área de conhecimento: Computabilidade e Modelos de Computação (CNPq: 1.03.01.01-1).

Simeone, A. L.; Seifert, J.; Schmidt, D.; Holleis, P.; Rukzio, E.; Gellersen ,H. A cross-device drag-and-drop technique. In: International Conference on Mobile and Ubiquitous Multimedia, 12., 2013, Luleå.

Tanenbaum, A. S.; Van Steen, M. Distributed Systems: Principles and Paradigms. 2. ed. Upper Saddle River: Prentice Hall, 2007.