UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DAVI AUGUSTO NEVES LEITE

MÓDULO DE RECONHECIMENTO DE GESTOS PARA AMBIENTES DE REALIDADE AUMENTADA

BAURU
Janeiro/2023

DAVI AUGUSTO NEVES LEITE

MÓDULO DE RECONHECIMENTO DE GESTOS PARA AMBIENTES DE REALIDADE AUMENTADA

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Assoc. Antonio Carlos Sementille

BAURU
Janeiro/2023

L533m

Leite, Davi Augusto Neves
Módulo de Reconhecimento de Gestos para Ambientes de Realidade Aumentada / Davi Augusto Neves Leite. -- Bauru, 2022
63 p. : il., fotos

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientador: Antonio Carlos Sementille

1. Ciência da computação. 2. Realidade aumentada. 3. Sistemas de computação interativos. 4. Processamento de linguagem natural (Computação). I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Davi Augusto Neves Leite

Módulo de Reconhecimento de Gestos para Ambientes de Realidade Aumentada

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Faculdade de Ciências, Campus Bauru.

Banca Examinadora

Prof. Assoc. Antonio Carlos Sementille
Orientador
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Profa. Dra. Simone das Graças Domingues Prado
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Prof. Dr. Kelton Augusto Pontara da Costa
Universidade Estadual Paulista "Júlio de Mesquita Filho"
Faculdade de Ciências
Departamento de Ciência da Computação

Bauru, 19 de Janeiro de 2023.

Dedico esse trabalho à minha família e meus amigos, pois sem eles nada na minha vida seria possível.

Agradecimentos

Agradeço principalmente a minha família, sobretudo minha mãe, avó e irmãs amadas a quem devo a minha vida.

Também, agradeço imensamente a todos meus amigos, os quais possibilitam com que eu continue vivendo e sorrindo na maioria dos dias. Em especial ao Eric e ao Vinicius, meus maiores salvadores durante todos esses últimos anos. Também para minha amiga Emanuelle que me ajudou muito em um dos piores momentos da minha vida e continua sendo incrível até hoje.

Obrigado Luis Morelli, Luiz Sementille, Giovani Candido, Gustavo Stahl e todos os outros por esses quatro anos de muita luta, risada e choro. Foi ótimo compartilhar o tempo com vocês enquanto durou.

Ainda, agradeço imensamente a meus calouros, os quais se tornaram uma grande parte de mim hoje em dia. Obrigado Arissa, Cassiano, Gustavo, Matheus, Marry, Nicole, Nicolas, Luan, Zastim, Dhiego, Henrique e a todos os outros que continuam fazendo parte deste meu livro chamado vida.

Agradeço também ao meu orientador Antonio Carlos Sementille, um dos melhores professores com quem tive o prazer de ter aula e que tem me ensinado e ajudado muito durante minha jornada universitária e na pesquisa científica.

Agradeço também a todos os professores que tive durante essa jornada, os quais me proporcionaram a me tornar um cidadão melhor na sociedade. Em especial, também, a professora doutoranda Juliana Feitosa, a qual tem sido uma inspiração para mim nessa parte acadêmica (e diversas outras coisas) desde meu início na faculdade em 2019.

Agradeço também a todos os membros maravilhosos do projeto Biblioteca Falada, principalmente sobre a equipe de T.I. a qual fiz parte. Obrigado Julia, você foi uma coordenadora incrível e continua sendo uma grande amiga pra mim. Obrigado também Inaê por todas as conversas e aprendizados, sempre será uma inspiração na minha vida.

Por fim, mas não menos importante, agradeço aos meus gatos, as bandas *Iron Maiden* e *Helloween*, todos os jogos *Souls-like*, *Sonic* e ao recente *Genshin Impact*, por me relaxarem e ajudarem nos piores e melhores momentos.

"So understand
Don’t waste your time always searching for those wasted years
Face up, make your stand
And realize you’re living in the golden years."
Wasted Years - Iron Maiden

Resumo

A Realidade Aumentada pode ser definida como a área responsável por enriquecer o mundo real por meio da combinação de objetos gerados por computador. Por conta disso, a Realidade Aumentada possui grande potencial para ser aplicada nas mais diversas áreas, como Educação, Medicina, Marketing e Área Industrial, uma vez que beneficia qualquer tarefa que necessite de acesso a uma informação que não teria normalmente. Contudo, a dificuldade na utilização da interatividade natural nas soluções recentes de *Head-Mounted Displays* voltadas para *smartphones* tem sido um dos principais obstáculos para a acessibilidade e popularização da Realidade Aumentada. Desta forma, era de interesse do projeto estruturar, implementar e avaliar um módulo reconhecedor de gestos de mão para ambientes de Realidade Aumentada. Os experimentos propostos consistiam na realização de ações diretas e indiretas do usuário, com relação aos objetos virtuais, em diferentes cenários da Realidade Aumentada, baseados em tarefas. Os resultados obtidos mostraram a viabilidade do módulo em aplicações de Realidade Aumentada com o uso de *Head-Mounted Displays Video See-Through* baseados em *smartphones* para tarefas em que a precisão não é um requisito fundamental.

Palavras-chave: Realidade Aumentada; Interação por Gestos de Mão; *Head-Mounted Displays*; *MediaPipe Hands*; *Vuforia*.

Abstract

Augmented Reality can be defined as the area responsible for enriching the real world through the combination of computer-generated objects. Because of this, Augmented Reality has great potential to be applied in the most diverse areas, such as Education, Medicine, Marketing, and the Industrial Area, since it benefits any task that requires access to information that it would not normally have. However, the difficulty in using natural interactivity in recent Head-Mounted Displays solutions for smartphones has been one of the main obstacles to the accessibility and popularization of Augmented Reality. In this way, it was in the interest of the project to structure, implement and evaluate a hand gesture recognizer module for Augmented Reality environments. The proposed experiments consisted of carrying out direct and indirect actions by the user, in relation to virtual objects, in different scenarios of Augmented Reality, based on tasks. The obtained results showed the feasibility of the module in Augmented Reality applications with the use of Smartphone-based Head-Mounted Video See-Through Displays for tasks where accuracy is not a fundamental requirement.
Keywords: Augmented Reality; Hand Gesture Interaction; Head-Mounted Displays; MediaPipe Hands; Vuforia.

Lista de figuras

Lista de quadros

Lista de abreviaturas e siglas

API	*Application Programming Interface*
AR	*Augmented Reality*
FOV	*Field Of View*
FHD	*Full High Definition*
GPU	*Graphics Processing Unit*
GR	*Gesture Recognition*
HMD	*Head-Mounted Display*
IHC	Interação Humano-Computador
IPD	*Interpupilary Distance*
ML	*Machine Learning*
MPAR	*MediaPipe Augmented Reality*
NUI	*Natural User Interface*
OST	*Optical See-Through*
RA	Realidade Aumentada
RGB	*Red Green Blue*
RGBD	*Red Green Blue Depth*
RV	Realidade Virtual
SLAM	*Simultaneous Localization and Mapping*
TOF	*Time of Flight*
VHM	*Virtual Hand Manager*
VST	*Video See-Through*
WSL	*Windows Subsystem for Linux*
XR	*Extended Reality*

Sumário

1 Introdução

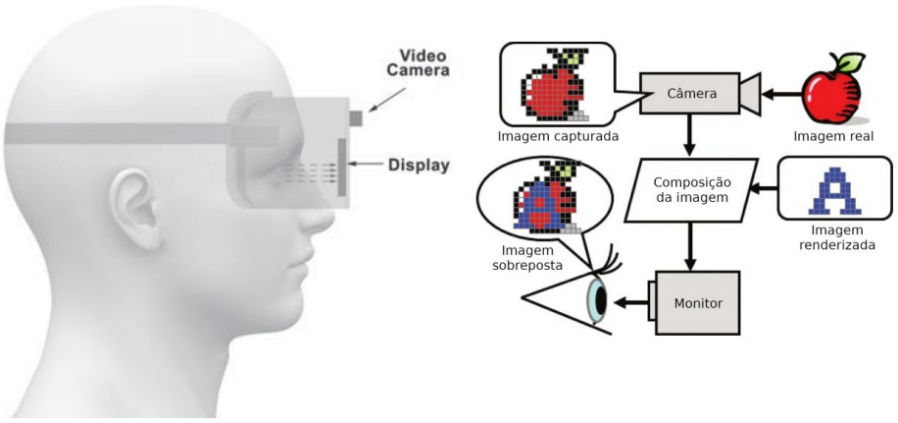
Os gestos, juntamente com a fala, consistem na principal forma de comunicação entre os seres humanos, abrangendo qualquer forma visual, tais como: movimentos de mão, expressões corporais e faciais. Al-Hammadi et al. (2020) definem que os gestos de mão contêm uma grande quantidade de informações, sendo fundamentais tanto na comunicação por linguagens de sinais quanto nos sistemas de Interação Humano-Computador (IHC).

Ainda de acordo com os autores, as soluções baseadas no reconhecimento automático de sinais por mão estão em alta demanda, uma vez que é possível desenvolver a interatividade com uma aplicação (ou dispositivo) por meio desse tipo de gestos. Um exemplo pode ser observado no sistema criado por Ameur, Khalifa e Bouhlel (2020), o qual utiliza um dispositivo não convencional, denominado "Leap Motion" (ULTRALEAP, 2020), para o rastreamento das mãos. A partir disso, torna-se possível manipular imagens médicas em um computador, durante um processo cirúrgico, por meio da interatividade “sem toques” e intuitiva com “gestos de mão”. Este tipo de interatividade é denominado de "baseada em visão". Neste tipo de interação não há necessidade de retorno tátil, como acontece na interatividade "baseada em contato".

Uma área importante da Computação e extremamente influenciada pela interatividade através de gestos da mão é a Realidade Aumentada (RA). De acordo com a definição de Azuma (1997), é possível resumir essa grande área em três pontos-chave simultâneos: combinação do conteúdo real e virtual; existência de interação em tempo real; e registro dos objetos reais e virtuais de forma coerente, no espaço 3D. Entende-se por virtual (ou digital) tudo aquilo que é construído por meios eletrônicos, como um computador ou celular.

Tendo em vista que os elementos digitais são combinados no mundo real, torna-se necessária a utilização de determinados dispositivos que possam prover essa combinação de maneira simultânea, como os chamados *Head-Mounted Display* (HMD). De acordo com Arnaldi, Guitton e Moreau (2018), os HMDs podem ser divididos em dois tipos, segundo sua arquitetura óptica: *Optical See-Through* (OST), em que as imagens reais e virtuais são combinadas a partir de lentes ópticas e, dessa forma, permitem uma visão natural direta; e *Video See-Through* (VST) em que o ambiente é percebido através de uma câmera e, portanto, a combinação dos elementos reais e virtuais são baseadas em imagens digitais. A Figura 1 ilustra o funcionamento básico dos HMDs VST, foco principal deste trabalho.

Figura 1 – Funcionamento dos HMDs VST



Fonte: Adaptada de Kiyokawa (2015).

A interatividade por meio do uso de interfaces naturais, como é o caso do reconhecimento de gestos, normalmente exige o uso de sensores de profundidade especiais, como o citado Leap Motion, os quais normalmente são projetados para serem acoplados em computadores do tipo *desktops* ou *notebooks*, mas tornam-se inviáveis de serem acoplados externamente em dispositivos móveis, tais como *smartphones* e *tablets*. Naturalmente existem alguns dispositivos móveis que já possuem sensores de profundidade embutidos, porém são ainda muito raros e de alto custo. Consequentemente, as pesquisas para a utilização dos *smartphones* neste âmbito têm crescido, conforme apontam Billinghurst, Clark e Lee (2015), uma vez que esses dispositivos, em sua grande maioria, possuem uma ou mais câmeras de alta resolução e componentes eletrônicos capazes de executar *softwares* para o processamento e síntese de imagens. Isso pode ser visto na popularização do dispositivo VR BOX 2, um HMD do tipo VST baseado no Google Cardboard, conforme mostrado na Figura 2.

Figura 2 – VR BOX 2: HMD VST de RA baseado em *smartphone*

Fonte: Elaborada pelo autor.

Recentemente, grandes empresas como a Google têm desenvolvido algumas soluções computacionais baseadas em Aprendizado de Máquina (ML) para o rastreamento de mãos em uma cena, como é o caso do *framework* MediaPipe Hands. O grande potencial deste *framework* tem se evidenciado em trabalhos recentes encontrados na literatura. Por exemplo, no trabalho de Rompapas et al. (2021), sugere-se a substituição do uso do sensor Leap Motion (e suas funções nativas) pelo uso de uma câmera RGB (*Red*, *Green*, *Blue*) comum e o MediaPipe Hands.

Além disso, outra abordagem de ML para rastreamento bastante consolidada na literatura é a baseada em marcadores fiduciais. Neste tipo, é exigido a existência de um ou mais objetos de referências na cena, como uma imagem impressa numa folha de papel, os quais são utilizados para estimar as posições dos objetos virtuais e do usuário no espaço da aplicação. As principais ferramentas de construção de aplicações de RA deste método são o ARToolKitX e o Vuforia.

1.1 Problemática

Grande parte das aplicações de RA está restrita às grandes empresas, uma vez que são necessários equipamentos de alto custo, como o "HoloLens 2"(MICROSOFT, 2022), para que se tenha o alinhamento correto dos objetos virtuais e reais no espaço 3D, conforme definido por Azuma (1997). Dessa forma, a alternativa encontrada a fim de popularizar essa tecnologia está, como mencionado anteriormente, no desenvolvimento de equipamentos HMD de baixo custo voltados para os *smartphones*. Entretanto, um problema decorrente desta solução consiste na dificuldade de utilização da interatividade natural, isto é, por meio de gestos de mão, uma vez que, durante o

reconhecimento de imagem, a inferência da profundidade do espaço 3D é dificultada pelo uso de apenas uma câmera RGB (comum nos *smartphones* atuais).

Ainda que o recente MediaPipe Hands ou o consolidado Vuforia possuam vários pontos positivos, como a não exigência de uma câmera de profundidade para realizar o rastreamento das mãos, ao contrário do Leap Motion, eles, infelizmente, não possuem funcionalidades nativas para o reconhecimento de gestos de mão, o que seria extremamente desejável na implementação de interação direta em ambientes de RA. Cabe, portanto, ao desenvolvedor destes tipos de aplicações implementar tais funcionalidades, o que não é uma tarefa trivial.

1.2 Justificativa

De acordo com Billinghurst, Clark e Lee (2015) em consonância com Arena et al. (2022), as aplicações de RA têm tido uma crescente e fundamental atuação nas áreas da Educação, Arquitetura, Medicina, Marketing, Turismo e Jogos Digitais. Contudo, como colocado anteriormente na seção 1.1, a dificuldade na utilização da interatividade natural nas soluções de HMD voltados a *smartphones* tem sido o principal obstáculo para a acessibilidade e popularização da RA.

Diante do contexto exposto, justifica-se o desenvolvimento de um módulo em *software* que forneça a funcionalidade do reconhecimento de gestos. A estruturação deste módulo, especificamente, levará em consideração a arquitetura de HMDs VST baseados em *smartphone* e visará dar suporte à interatividade baseada em visão para aplicações de RA, no espaço próximo ao corpo do usuário (espaço peripessoal).

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo principal deste projeto, portanto, consiste em estruturar, implementar e validar, um módulo em *software* capaz de reconhecer gestos de mão e que possa ser utilizado para interação em aplicações de RA no espaço peripessoal do usuário. O desenvolvimento deste módulo levará em conta a abordagem de ML para rastreamento de objetos e será direcionado, especificamente, aos HMDs VST baseados em *smartphone*.

1.3.2 Objetivos Específicos

- Determinar alguns dos gestos mínimos de mão para a interatividade básica em ambientes de RA.

- Estruturar e implementar um módulo, em termos de *software*, capaz de reconhecer gestos de mão que possam ser utilizados para interação em aplicações de RA baseadas em *smartphones*.
- Realizar testes em diferentes contextos da RA e analisar os resultados obtidos, por meio da utilização de métricas objetivas e quantitativas.

1.4 Organização do Trabalho

O presente trabalho está estruturado da seguinte forma: no Capítulo 2 tem-se a definição teórica e principais informações sobre a RA e de outros temas considerados mais importantes para o trabalho. Já o Capítulo 3 descreve a respeito do desenvolvimento do módulo, com relação a sua abordagem, arquitetura lógica e funcionamento básico. No Capítulo 4, são descritos os experimentos e discutidos os resultados obtidos, levando em consideração o módulo desenvolvido. Por fim, no Capítulo 5 são apresentadas as considerações finais e algumas possibilidades de trabalhos futuros.

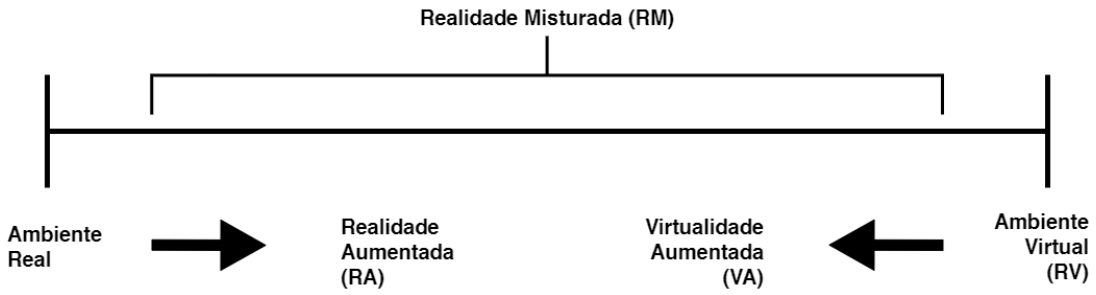
2 Fundamentação Teórica

Neste capítulo, os conceitos considerados mais importantes para este trabalho são apresentados, tais como: Realidade Aumentada, Interação Humano-Computador via Gestos de Mão e Espaço Peripessoal.

2.1 Realidade Aumentada

Para compreender o conceito de RA, é necessário entender o chamado “continuum de virtualidade”: um diagrama que estabelece as diferenças entre os ambientes real e virtual, apresentando seus limites por meio das definições de um objeto real e um virtual. Proposto inicialmente por Milgram e Kishino (1994), é possível visualizar que a RA está localizada na região da centro-esquerda, na chamada “área de transição” entre o mundo totalmente real para o mundo virtual, como visto na Figura 3. É importante mencionar que, ainda que esse conceito esteja sendo revisitado e discutido por diversos autores, como Skarbez, Smith e Whitton (2021), de tal forma a atualizá-lo com base nas tecnologias vigentes, o “Continuum de Milgram” (como também é chamado) é um conceito-chave amplamente utilizado para compreender as caracterizações dos ambientes real e virtual.

Figura 3 – Continuum de Virtualidade



Fonte: Adaptada de Milgram e Kishino (1994).

Desta forma, Arnaldi, Guitton e Moreau (2018) definem a RA como a capacidade dada a um ou mais usuários de realizar um conjunto de tarefas a partir do enriquecimento do ambiente real por meio da adição de objetos virtuais 2D ou 3D e, para isso, leva-se em conta dois principais fatores: imersão, ou seja, a percepção do usuário de que os objetos virtuais estão fisicamente presentes no ambiente real; e a interatividade, dada como a capacidade do usuário de modificar os objetos virtuais por meio de ações reais. Complementarmente, Hounsell, Tori e Kirner (2020) definem que

a arquitetura genérica de um sistema de RA é dividida em três módulos: entrada, a qual é responsável por capturar a cena real e possibilitar o rastreamento; processamento, associada ao tratamento de renderização correta dos objetos virtuais na cena e de processamentos relacionados à interação; e saída, a qual está relacionada com a apresentação final dos elementos reais e virtuais, por meio de um *display*.

Por fim, evidencia-se, então, que a importância de realização de pesquisa nesta área, conforme abordado por Hounsell, Tori e Kirner (2020), Barfield (2017), Billinghurst, Clark e Lee (2015) e Arena et al. (2022), está diretamente relacionada na vasta área de aplicabilidade em que a RA pode se inserir, como as já citadas Educação, Arquitetura, Medicina, Marketing, Turismo e, principalmente, na Área Industrial. Isso é justificado por meio da própria definição de RA, uma vez que os objetos virtuais devem ser combinados no mundo real, isto é, é possível inserir e mostrar informações as quais o usuário normalmente não teria acesso, infere-se, portanto, que “atividade humana que necessita de acesso à informação para melhor ser executada, pode se beneficiar da RA” (HOUNSELL; TORI; KIRNER, 2020).

2.2 Interação Via Gestos de Mão

A IHC, como explica Petrick (2020), é uma área interdisciplinar pautada a compreender e melhorar o relacionamento entre as ações do ser humano com um computador. Para tanto, é levado em conta as diferentes partes do corpo de uma pessoa com as diferentes entradas e saídas do computador. Contudo, como demonstra a autora e é colocado também por Sinha, Shahi e Shankar (2010), a área ainda não possui uma definição formal de fato, uma vez que os estudiosos abordam a respeito de “interfaces de computador”, "entrada-saída de dispositivos”, “sentidos humanos com tecnologia”, “computadores vestíveis”, “interação homem-máquina”, dentre outras alcunhas; ao contrário do uso explícito de “interação humano-computador”. Diante disso, para fins de caracterização e levando em conta a área da Ciência da Computação, este trabalho se baseia na seguinte definição: “A Interação Humano-Computador é uma disciplina preocupada com o projeto, avaliação e implementação de sistemas de computação interativos para uso humano e com o estudo dos principais fenômenos que os cercam” (SINHA; SHAHI; SHANKAR, 2010).

Diante disso, e levando em conta a definição de RA tratada anteriormente, a interação do usuário com o sistema é fundamental e indispensável para que haja a existência, de fato, de uma RA. Especificamente, Reifinger et al. (2007) explica que, para uma aplicação de RA atingir a sua totalidade imersiva, os sistemas de entrada e saída devem ser adaptados com a realidade do usuário, isto é, é necessário que compreendam a chamada “linguagem natural do ser humano”, como a fala e os gestos.

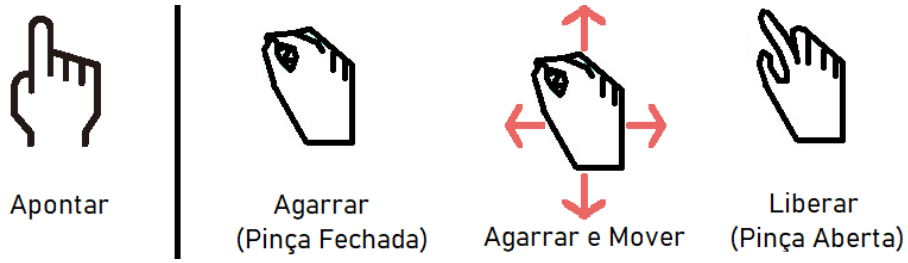
Neste contexto, Ghazwani e Smith (2020) e Aliprantis et al. (2019) explicam que as “interfaces naturais de usuário” (NUI) consistem num sistema que possibilita o uso de movimentos naturais como método de entrada na interação com os objetos virtuais no RA. Em outras palavras, é possível utilizar as informações obtidas através da fala ou de gestos para realizar as ações no sistema. Para tanto, Aukstakalnis (2017) em consonância com Ghazwani e Smith (2020), esclarecem que essas interfaces comumente estão associadas ao uso de aparelhos externos, como a luva de dados para a mão ou de sensores de captura para o corpo, de tal forma que a precisão das informações obtidas seja valorizada. Contudo, isto torna-se inviável para o usuário na prática, uma vez que esses equipamentos, em sua grande maioria, são custosos financeiramente.

Em face do exposto, os gestos de mão tornam-se fundamentais para a realização de interação em sistemas de RA com NUI. Conforme explica Pinho et al. (2020) em concordância com Peddie (2017), o grande ponto positivo da utilização deste tipo específico de interação consiste em se aproveitar do conhecimento intuitivo do próprio usuário, de tal forma a alcançar a facilidade e naturalidade. Diante disso, a literatura se divide em duas abordagens diferentes para o reconhecimento dos gestos de mão em um sistema, como apresenta Khurshid et al. (2022): baseadas no uso de luvas de dados, como citadas anteriormente; e as baseadas no uso de câmeras. Com relação à segunda, Oudah, Al-Naji e Chahl (2020) em conformidade com Khurshid et al. (2022) explicam que o uso da visão computacional, isto é, aplicação de técnicas computacionais para segmentação e detecção de pontos característicos da mão, como cor da pele, esqueleto, movimentação e modelo 3D, tem crescido na última década, em conformidade com o avanço no desenvolvimento das câmeras de vídeo comum (RGB), das câmeras de profundidade (RGBD, como o Kinect) e de sensores como o “tempo de voo” (TOF), os noturnos e termais. Somado a isso, Aliprantis et al. (2019) constata a importância dos dispositivos móveis neste contexto, uma vez que as câmeras, sensores e processadores gráficos embutidos nesses aparelhos possibilitam, ainda que em estágio primitivo se comparado aos computadores de mesa, a aplicação de visão computacional sem a adição de outros equipamentos, de tal forma que seja possível construir aplicações de RA de maneira portátil.

Por fim, é importante eleger quais seriam os gestos mínimos para uma aplicação de RA. Um trabalho importante sobre a temática foi realizado por Piumsomboon et al. (2013). Em seus estudos, os autores elegeram 44 gestos para 40 ações nos cenários de RA. Dentre eles, é possível eleger e adaptar alguns para esse trabalho, com base em suas ações: “apontar”, utilizado para seleção indireta de objetos (sem colisão); “pinça fechada”, relacionado com a seleção direta de objetos (com colisão), e sua variação “pinça fechada e mover” atrelada ao movimento; “pinça aberta”, empregado para encerrar a operação da “pinça fechada”, liberando o objeto dos dedos do usuário.

Esses gestos podem ser vistos na Figura 4.

Figura 4 – Gestos mínimos presentes no sistema



Fonte: Elaborada pelo autor.

2.3 Espaço Peripessoal

O espaço peripessoal, de acordo com Bufacchi e Iannetti (2018) e Hunley e Lourenco (2018), pode ser definido como o espaço máximo alcançável pelos braços humanos. Em outras palavras, exprime a região em que os objetos podem receber uma ação direta de um ser humano. Ainda que isso varie de pessoa a pessoa, a distância média do comprimento deste espaço se estende em torno de 1m.

3 Materiais e Métodos

Este capítulo dispõe a respeito do processo de construção do módulo, tal como os materiais, modelagem, arquitetura do sistema e detalhamento do desenvolvimento. Especificamente, utilizou-se de duas abordagens distintas para a construção do módulo. A primeira, com o uso MediaPipe Hands, apresentou problemas no desenvolvimento e, por conta disso, tornou-se necessário a reestruturação integral do módulo levando em conta os serviços do consolidado Vuforia. Esta segunda abordagem foi a considerada nos experimentos propostos na seção 4.1 e, portanto, deve ser tomada como principal neste trabalho.

Além disso, por se tratar de um sistema complexo, seguiu-se os conceitos e métodos de modelagem descritos por Pressman e Maxim (2021) em que a chamada "modularidade" deve ser baseada em restrições. Em outras palavras, o *software* deve ser dividido em partes (como contêineres), cada qual com sua representatividade funcional e de tal modo que a interconexão entre elas seja relativamente simples. Adicionalmente, como enfatizam os autores, uma arquitetura simboliza a representação operacional do sistema, de forma que seja possível analisar a efetividade do projeto.

3.1 Materiais Gerais

Considerando o contexto citado, escolheu-se as seguintes ferramentas gerais do projeto, ou seja, aquelas que estão presentes em ambas abordagens:

Software

- Sistemas Operacionais: Windows 11 para *desktop* e Android 11 para o *smartphone*;
- Ambiente de Desenvolvimento Integrado: Microsoft Visual Studio Code;
- Motor de Jogos: Unity3D v2021.3.3f1;
- Linguagem de Programação: C#.

Hardware

- Notebook pessoal Lenovo Ideapad 330-15IKB com: processador Intel Core i7-8550U, HDD WD Blue WD10SPZX de 1TB, SSD Crucial BX500 de 240GB, 12 GB DDR4 de Memória RAM e placa de vídeo NVIDIA GeForce MX150 (2 GB GDDR5 de memória);

- *Smartphone* POCO X3 NFC com: 6 GB de Memória RAM, processador Qualcomm Snapdragon 732G, GPU Adreno 618 e quatro câmeras principais;
- HMD VST: VRG PRO.

Os principais materiais gerais do projeto serão descritos a seguir, como o Unity3D, o *smartphone* POCO X3 NFC e o HMD VST VRG PRO. Destaca-se que os materiais adicionais de cada abordagem serão descritos respectivamente em suas seções posteriores.

3.1.1 Unity3D

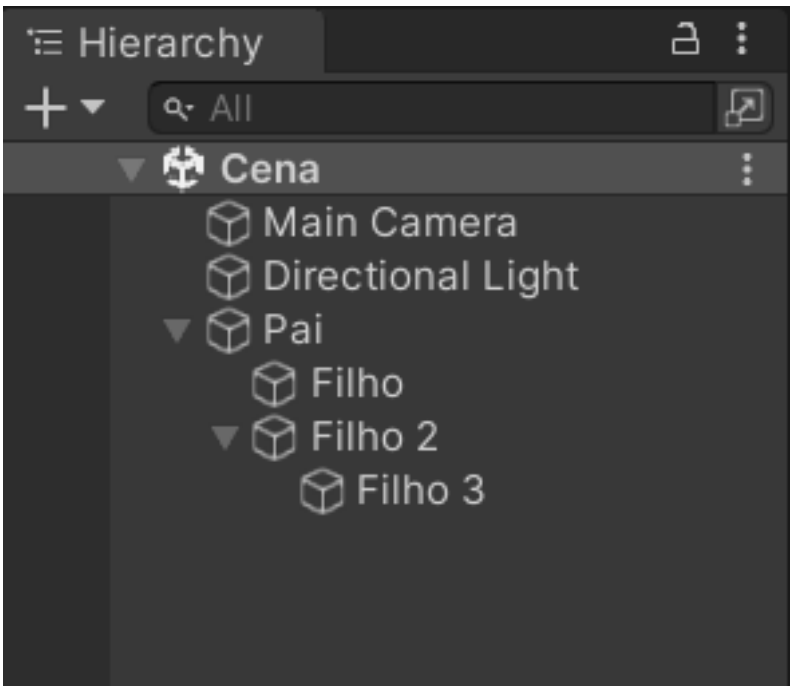
De acordo com a própria desenvolvedora Technologies (2022), a Unity (ou Unity3D) é uma plataforma de desenvolvimento de aplicações interativas com gráficos 3D ou 2D, em tempo real, e com suporte multiplataforma. Ademais, é possível elencar as seguintes principais características incluem: detém uma série bibliotecas para computação gráfica, denominadas de pacotes, como as de criação e manipulação de formas geométricas básicas; suporte para importação de modelos geométricos complexos, criados em *softwares* de modelagem (como Autodesk ou Blender); estrutura baseada em componentes, isto é, conta com suporte da criação de *scripts* para objetos e gerenciadores, por meio da linguagem majoritária C#; dentre outras.

Atualmente, desempenha um papel fundamental para a popularização da RA por possibilitar aos desenvolvedores que criem suas aplicações de forma eficaz. Em outras palavras, a Unity3D propicia um ambiente com a capacidade de multiplataforma, a qual exprime que um projeto feito para determinada plataforma, como por exemplo o Android nos *smartphones*, possa ser exportado sem muitas dificuldades para outras plataformas, como o iOS ou Windows. Além disso, a vasta gama de pacotes terceirizados disponíveis tanto em sua loja oficial quanto em outras plataformas famosas da comunidade, como o GitHub, e do seu recentemente lançado kit de desenvolvimento de *software* (SDK) XR para criação de aplicações da Realidade Estendida (como a RA), tem tornado esse motor de jogos cada vez mais consolidado para a criação de aplicações de RA.

Uma das funções importantes oferecidas pela Unity3D consiste em organizar o ambiente virtual por meio de uma estrutura hierárquica. Ou seja, é possível dispor os objetos virtuais de acordo com sua funcionalidade e prioridade na cena, o qual auxilia o chamado *pipeline* gráfico (renderização) e na organização do projeto. Dentro da ferramenta, essa função pode ser vista na chamada “Janela de Hierarquia” (ou *Hierarchy*), conforme é mostrado na Figura 5. Nota-se, por exemplo, que os objetos “Filho” e “Filho 2” são filhos diretos de “Pai”, ao passo que “Filho 3” é filho direto de “Filho 2” e indireto de “Pai”. Essa disposição dos objetos permite com que o mais alto da hierarquia comanda diretamente seus filhos. Por exemplo, caso o objeto “Pai” sofresse

uma operação de rotação, todos os seus filhos diretos (“Filho” e “Filho 2”) e indiretos (“Filho 3”) sofreriam a mesma operação.

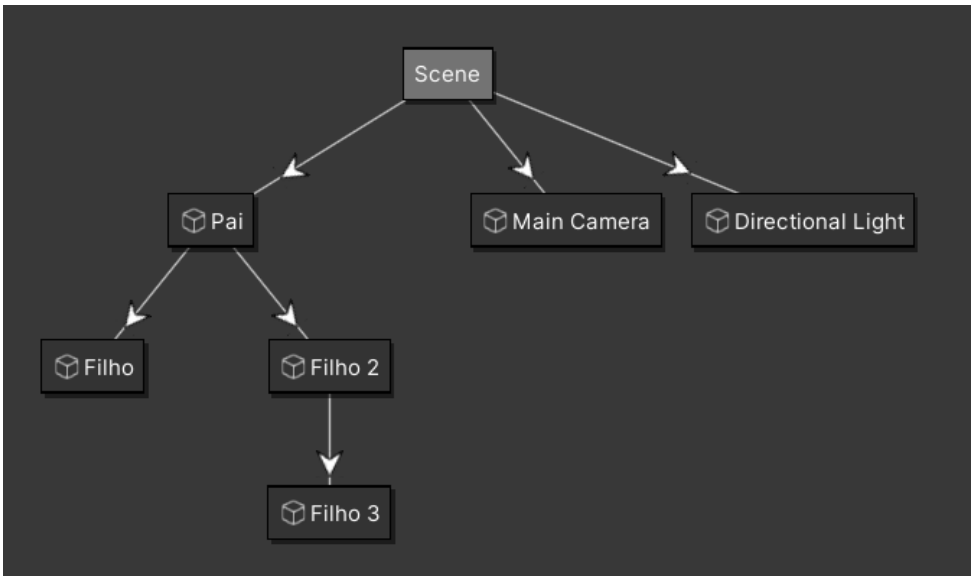
Figura 5 – Exemplo de cena na Janela de Hierarquia da Unity3D



Fonte: Elaborada pelo autor.

Especificamente, cada cena construída na Unity3D está associada ao seu respectivo grafo de cena, o qual facilita a visualização da hierarquia e torna possível visualizar a disposição dos objetos para o *pipeline* gráfico. Em um grafo de cena, cada nó representa um objeto e cada aresta representa a relação de parentesco entre dois objetos. Por exemplo, o grafo de cena apresentado na Figura 6 é referente a cena apresentada na Figura 5.

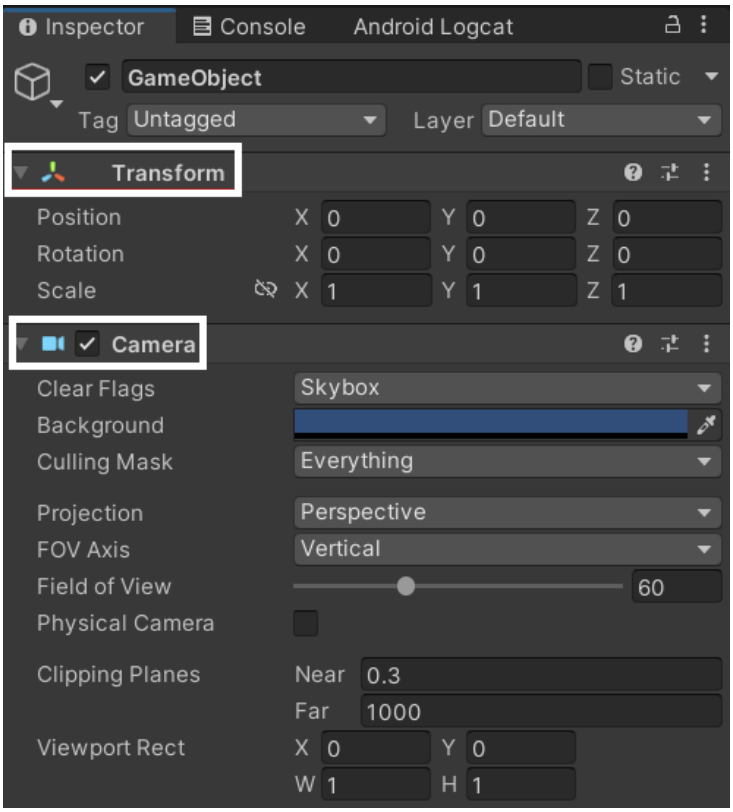
Figura 6 – Exemplo de grafo de cena da Unity3D



Fonte: Elaborada pelo autor.

Por fim, destaca-se dois conceitos fundamentais utilizados dentro da ferramenta: *GameObjects* e componentes. Os *GameObjects* são considerados todos os objetos de cena e servem como receptáculo para os chamados componentes, os quais consistem em estruturas que desempenham e/ou agrupam uma ou mais funcionalidades no sistema. Como exemplo de componentes, tem-se os colisores (*Collider*), *scripts*, simuladores de física (*Rigidbody*), dentre outros. Os componentes, dentro da ferramenta, podem ser vistos na “Janela de Inspeção” (ou *Inspector*), como indicado em vermelho na Figura 7. Especificamente, o componente de transformada (*Transform*) é necessário em todo *GameObject* criado, uma vez que ele é responsável por gerenciar tanto a posição, rotação e escala de um objeto quanto garantir a existência da hierarquia de cena por meio da conexão entre objetos. Em outras palavras, o *Transform* desempenha a função de aresta em um grafo de cena na Unity3D.

Figura 7 – Exemplo de *GameObject* com componentes *Transform* e *Camera*



Fonte: Elaborada pelo autor.

3.1.2 *Smartphone* POCO X3 NFC

O *smartphone* era um dos equipamentos fundamentais para a execução do trabalho, uma vez que o módulo seria testado diretamente nele, tendo em vista a arquitetura de funcionamento dos HMDs VST e das abordagens de ML para rastreamento de objetos em cena. Para tanto, utilizou-se o POCO X3 NFC, considerado um aparelho com “especificações intermediárias” do mercado, de acordo com Santos (2021) e Alecrim (2020). É possível visualizá-lo na Figura 8. Dentre as especificações estabelecidas pela empresa POCO (2020), ressaltam-se seis para as aplicações de RA: suporte ao ARCore (SDK para RA no Android); memória RAM LPDDR4X de 6GB; processador Qualcomm Snapdragon 732G com poderosos oito núcleos de 2.3GHZ cada; GPU Adreno 618 com até 800MHz de velocidade; *display* de nanocristal com tamanho de 6.67”, resolução de 2400 x 1080 FHD (*Full High Definition* - Alta Definição Total) e 120Hz de taxa de atualização; e quatro câmeras traseiras.

Figura 8 – Visão frontal e traseira do POCO X3 NFC

Fonte: Elaborada pelo autor.

Com relação a última especificidade, isto é, a respeito das câmeras, é importante destacar sua importância para o projeto, uma vez que o ambiente real capturado em um cenário de RA deve ser o de maior qualidade possível. Diante disso, é destacado pela POCO (2020) que a câmera RGB principal possui 64MP, sendo a melhor, com relação às outras três, para a captura de ambientes próximos do usuário e, portanto, para aplicações de RA no espaço peripessoal. Outrossim, ela “registra imagens com ótima definição, cores vívidas, mas não excessivamente saturadas, e controle decente de ruído” (ALECRIM, 2020).

3.1.3 HMD VST VRG PRO

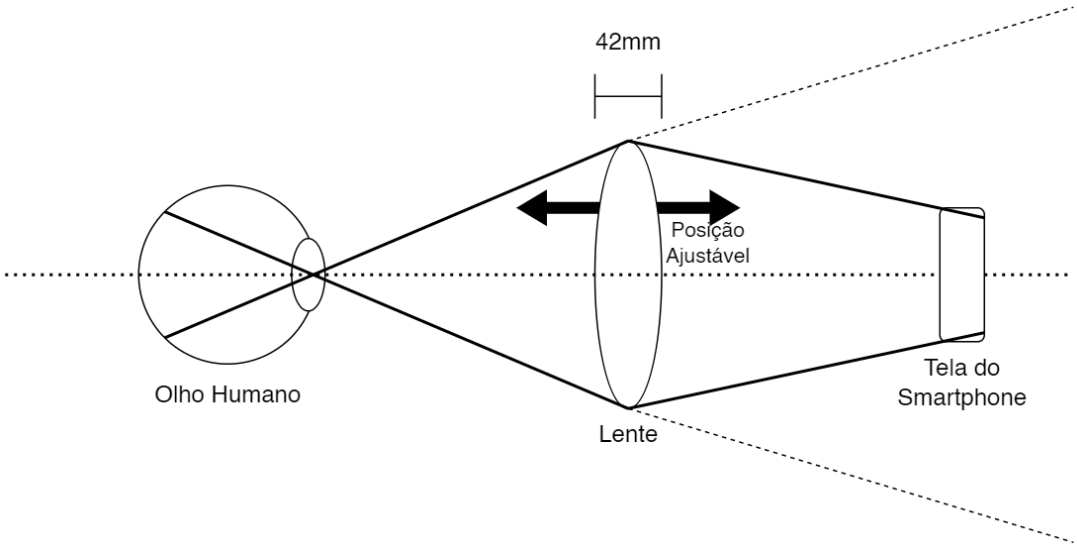
Considerando o tamanho de tela do POCO X3 NFC, tornou-se necessário a aquisição do VRG PRO que, assim como o VR BOX 2, é um HMD VST baseado no Google Cardboard e de baixo custo, podendo ser utilizado para experiências em Realidade Virtual (RV) ou RA de maneira direta. O equipamento pode ser visto na Figura 9.

Figura 9 – Visão frontal do VRG PRO

Fonte: Elaborada pelo autor.

Com relação às suas características, especialmente a sua arquitetura interna, o HMD possui uma lente para cada olho, com 42mm de espessura e comprimento focal de 53.3mm, e suporte para ajuste da posição da lente e da distância interpupilar (IPD). A IPD pode ser ajustada numa faixa de 60 a 75mm. Além disso, seu chamado “suporte para miopia” consiste no ajuste das lentes em até 800°. Por fim, possui um campo de visão (FOV) de 120° na horizontal, sendo maior do que os 80° do VR BOX 2. É possível visualizar sua arquitetura interna na Figura 10.

Figura 10 – Arquitetura interna do VRG PRO



Fonte: Elaborada pelo autor.

3.2 Abordagem Inicial: MediaPipe Hands

Esta seção apresenta o processo de construção do módulo, por meio da abordagem do MediaPipe Hands. Ou seja, são descritos os materiais adicionais necessários, a modelagem e arquitetura do sistema, os detalhes no desenvolvimento e os contratempos encontrados.

3.2.1 Materiais Adicionais

Adicionalmente aos materiais gerais, tornou-se necessário a utilização dos seguintes materiais para essa abordagem:

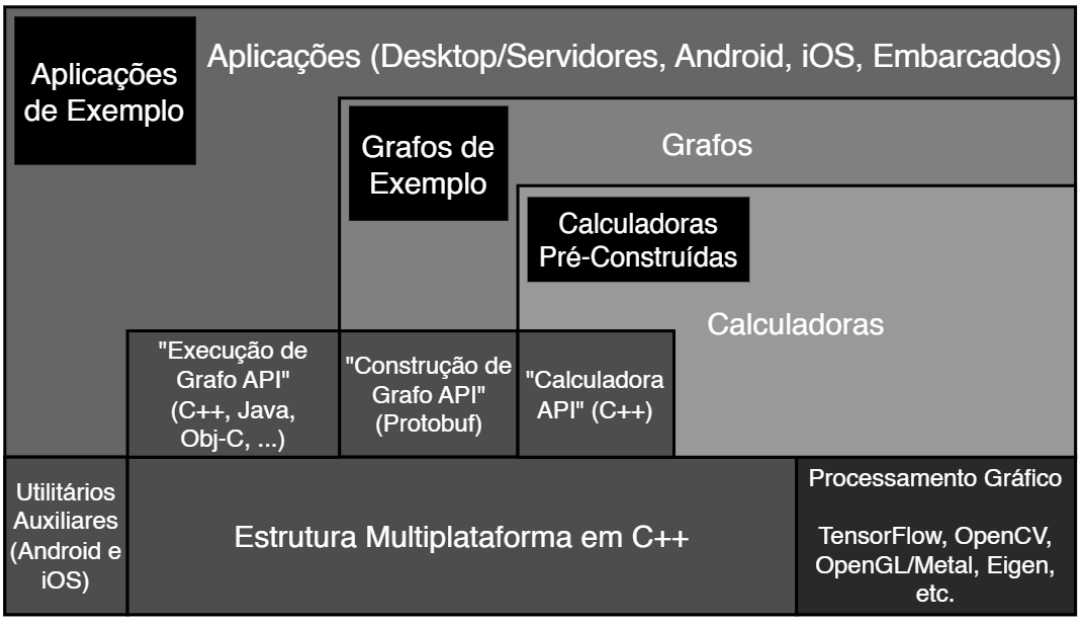
Software

- Biblioteca de RA: AR Foundation v5.0.3 (ARCore v1.31) para Unity3D;
- *Framework* de ML adaptado para Unity3D: “MediaPipeUnityPlugin” v0.10.1 (Google MediaPipe v0.8.10).

3.2.1.1 MediaPipe Hands

Dentro do contexto situado, uma estrutura recentemente publicada tem chamado a atenção dos pesquisadores da área de RA. O MediaPipe se trata de um *framework* de código-aberto, publicado oficialmente por Lugaresi et al. (2019), com a finalidade de ofertar a criação de *pipelines* (ou segmentação de instruções), baseadas em aprendizado de máquina, para processar dados como áudio e vídeo. O grande diferencial desta ferramenta está em sua arquitetura, a qual possibilita o funcionamento estável e otimizado em dispositivos móveis e aparelhos embarcados. De acordo com Yong (2019), um dos pesquisadores e engenheiros da Google, é possível dividi-la em quatro principais partes, a partir de sua base: uma camada multiplataforma escrita em C++, a qual abrange as ferramentas de reconhecimento de imagem e de comunicação direta com a GPU de um dispositivo, como OpenCV, OpenGL e TensorFlow; uma camada denominada de “Calculadora”, a qual abstrai a camada anterior por meio da construção de uma estrutura de dados específica, denominada “Calculadora”, e cuja função está associada a designação de tarefas para processamento; uma camada de “Construção de Grafos” com a utilização do Protobuf, responsável por fornecer uma interface baseada em grafos para a comunicação entre as Calculadoras; e, por fim, a camada de “Execução dos Grafos”, responsável por fazer a comunicação direta com as aplicações, isto é, trata-se de um módulo que realiza a execução dos “Grafos” e é manipulado em linguagens de programação de alto nível, como Java, Objective-C, JavaScript e Python. A Figura 11 ilustra a arquitetura do MediaPipe num geral.

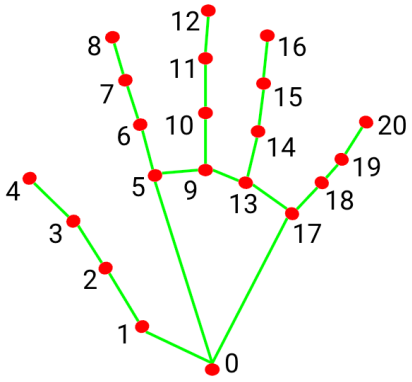
Figura 11 – Arquitetura interna do MediaPipe



Fonte: Adaptada de Yong (2019).

Especificamente, uma solução revelada por Zhang et al. (2020) diz respeito ao rastreamento de mãos, em tempo real, a partir de apenas uma câmera RGB e com o foco nas aplicações de RA. Denominada de MediaPipe Hands, é possível inferir até 21 pontos de referência 2.5D para cada mão em um único *frame* processado, como indicado na Figura 12. É importante ressaltar que os pontos inferidos são do tipo 2.5D uma vez que o eixo de profundidade (z) é relativo ao ponto 0 (pulso) e calculado a partir da largura e altura do quadro processado.

Figura 12 – Pontos de rastreamento de mão inferidos pelo MediaPipe Hands



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Fonte: Google (2020).

Para tal propósito, essa solução recorre simultaneamente de dois módulos da biblioteca TensorFlow: o detector de palma (*palm detection*) e o marco de mão

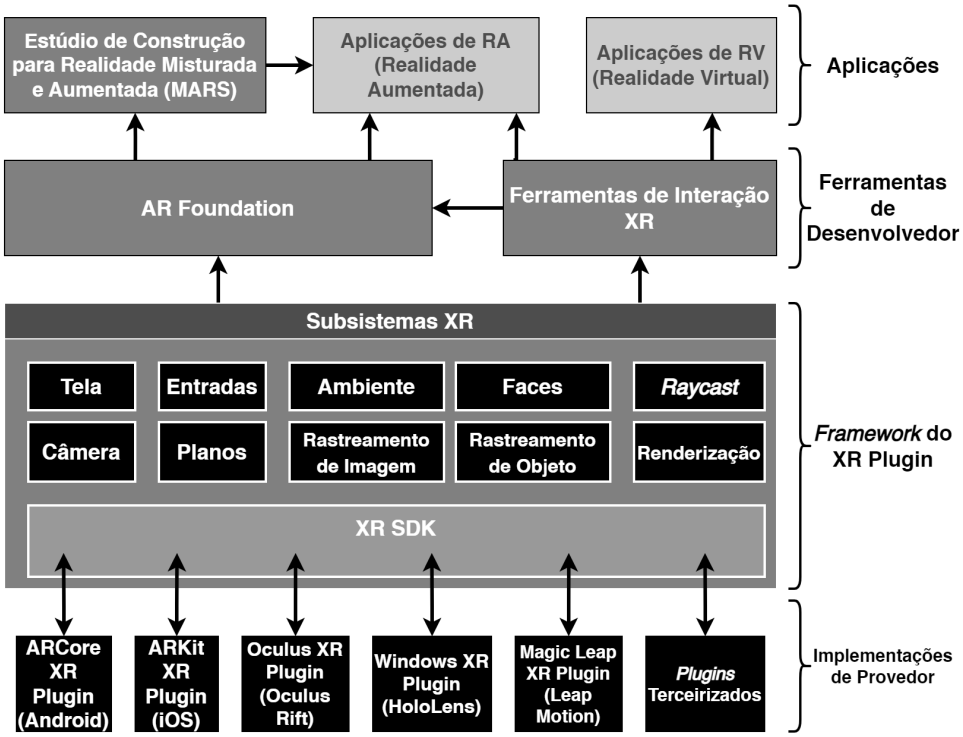
(*hand landmark*). Zhang et al. (2020) exprimem que, a partir de modelos treinados, o *palm detection* verifica o quadro da imagem por completo e infere a chamada caixa delimitadora de mão (*hand bounding box*), a qual por sua vez simboliza a região delimitada de uma mão encontrada no quadro. Por conseguinte, o *hand landmark* opera somente nesta região delimitada e, também a partir de modelos treinados, infere os 21 pontos 2.5D de alta fidelidade. Por conta desta arquitetura e do suporte às Interfaces de Programação de Aplicativos (API) de gráficos móveis, o MediaPipe Hands pode ser utilizado com estabilidade e otimização em *smartphones*, tornando-o atrativo para o uso em sistemas de RA móveis, como indicam os próprios autores.

Contudo, não existe até o momento uma publicação oficial do MediaPipe Hands para a Unity3D. Diante disso, tornou-se necessário a adaptação do *framework* para seu devido uso com o motor de jogos. Inicialmente, a solução avaliada consistia em uma adaptação do código original, escrito na linguagem de programação C++, para um formato de *plugin* nativo da Unity, seguindo as recomendações de construção dadas pela Technologies (2022). Felizmente, Nishida (2022) realizou essa adaptação, de forma estável, e tornou-a pública via GitHub, juntamente com uma série de instruções de uso e soluções suportadas (como a de rastreamento de mão), sempre acompanhando as atualizações advindas do MediaPipe oficial.

3.2.1.2 AR Foundation para Unity3D

O AR Foundation é uma estrutura contida dentro do XR SDK da Unity3D que permite ao desenvolvedor criar aplicações de RA para as mais diversas plataformas existentes, com alvo para os dispositivos móveis. Para isso, conforme explica a Technologies (2022) em sua documentação oficial, este *framework* se utiliza dos recursos de RA providos pelo próprio sistema operacional de destino, via *plugin* específico para Unity3D, como o ARCore do Android ou ARKit do iOS, como indicado na arquitetura de funcionamento do XR SDK presente na Figura 13.

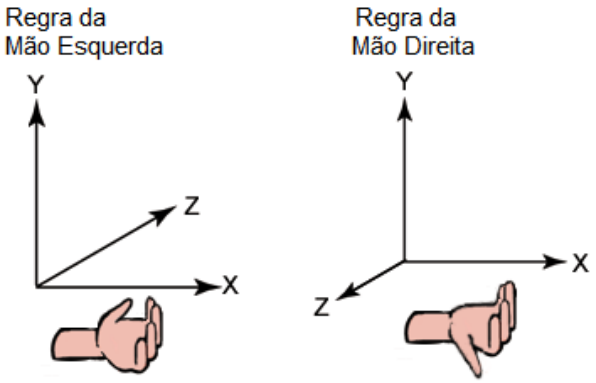
Figura 13 – Arquitetura de funcionamento do XR SDK



Fonte: Adaptada de Technologies (2022).

Diante disso, é importante ressaltar a existência desses *plugins* específicos presentes na camada da base, uma vez que são eles os responsáveis por realizar a chamada “conversão dos sistemas de coordenadas” entre as coordenadas obtidas de um sistema operacional de destino, como o ARCore do Android, para as coordenadas da Unity3D, e vice-versa, durante a execução da aplicação. Por exemplo, o ARCore, de acordo com a Google (2022), utiliza-se de um sistema de coordenadas baseado na chamada “regra da mão direita”: a coordenada X é incrementada à direita; a coordenada Y é a vertical e incrementada para cima; e a coordenada Z é incrementada para trás. Neste cenário, o *plugin* realiza a conversão das coordenadas obtidas, por uma câmera ou outra entrada, para o sistema utilizado pela Unity3D, baseado na “regra da mão esquerda”: a coordenada X é incrementada à direita; a coordenada Y é a vertical e incrementada para cima; e a coordenada Z é incrementada para frente. Após a Unity3D processar os dados possíveis na aplicação, as coordenadas são retornadas para o *plugin*, o qual realiza a conversão para o ARCore e, desta forma, possibilitando o aparelho com Android executar corretamente a aplicação de RA. É possível visualizar as diferenças entre os dois sistemas na Figura 14. A principal diferença, diante do exposto, está na coordenada Z, relacionada a profundidade da cena, considerada de extrema importância no contexto do projeto, uma vez que é necessário conhecer corretamente o seu valor para o adequado registro e rastreamento dos objetos virtuais.

Figura 14 – Diferenças entre os sistemas de coordenadas baseados na regra da mão esquerda e direita



Fonte: Adaptada de Turner e Coulter (2021).

Além disso, evidencia-se os seguintes principais recursos do AR Foundation para os aparelhos Android com suporte ao ARCore: rastreamento do dispositivo, como sua posição e rotação, em um espaço físico; detecção e rastreamento de planos e pontos característicos; suporte para oclusão e para profundidade (Depth API do ARCore); dentre outros. A lista completa de funcionalidades, separados por *plugin*, pode ser vista na Figura 15. Nota-se principalmente que as plataformas móveis (Android e iOS) não possuem a funcionalidade de reconhecimento de gestos para RA, foco principal de construção deste trabalho.

Figura 15 – Funcionalidades previstas no AR Foundation, por plataforma de destino

Unity AR Foundation Funcionalidades Suportadas

Funcionalidades	ARKit	ARCore	Magic Leap	HoloLens
Passagem por Vídeo	O	O		
Rastreamento de Dispositivo	O	O	O	O
Raycast	O	O	O	O
Rastreamento de Plano	O	O	O	
Pontos de Referência	O	O	O	O
Detecção de Pontos de Nuvem	O	O		
Gestos			O	O
Rastreamento de Face	O	O		
Rastreamento de Imagem 2D	O	O		
Rastreamento de Objeto 3D	O			
Texturização do Ambiente	O			
Meshing			O	O
Rastreamento 2D e 3D de Corpo	O			
Segmentação Humana e Oclusão	O			
Ambiente Colaborativo	O			

Fonte: Adaptada de Technologies (2022).

3.2.2 Métodos

Nesta seção, são apresentadas as modelagens e arquiteturas do módulo com a abordagem do MediaPipe Hands, bem como o desenvolvimento do mesmo. Destaca-se a existência de contratempos e dificuldades durante a elaboração deste módulo.

3.2.2.1 Modelagem e Arquitetura do Sistema

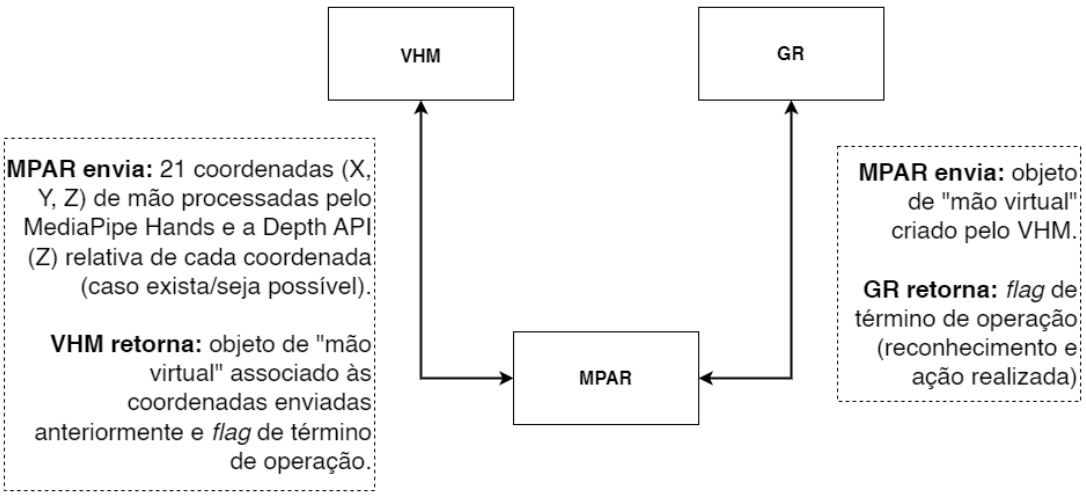
Diante do exposto e dos materiais descritos, o módulo de reconhecimento de gestos nesta abordagem foi estabelecido em três principais partes: a classe “MPAR” (MediaPipe Augmented Reality – MediaPipe para Realidade Aumentada); a classe “VHM” (Virtual Hand Manager – Gerenciador de Mão Virtual); e a classe “GR” (Gesture Recognition – Reconhecimento de Gestos).

Especificamente, descreve-se cada classe da seguinte maneira:

- **MediaPipe Augmented Reality (MPAR)**: responsável por configurar e inicializar todas as estruturas necessárias, como o MediaPipe Hands e o AR Foundation, e integrá-las inicialmente, de tal maneira a obter o rastreamento de mãos pelos quadros obtidos da câmera de RA;
- **Virtual Hand Manager (VHM)**: consiste, principalmente, em recuperar as coordenadas obtidas pelo MPAR e criar um objeto de “mão virtual”, associando os valores obtidos com as posições do objeto na cena;
- **Gesture Recognition (GR)**: realiza o reconhecimento de gestos propriamente dito, por meio da distância relativa ao ponto 0 (pulso) entre os pontos característicos de um gesto (como o indicador para o gesto “apontar) e, após isso, infere uma ação sobre os objetos virtuais no sistema, com base nos quatro gestos propostos neste trabalho.

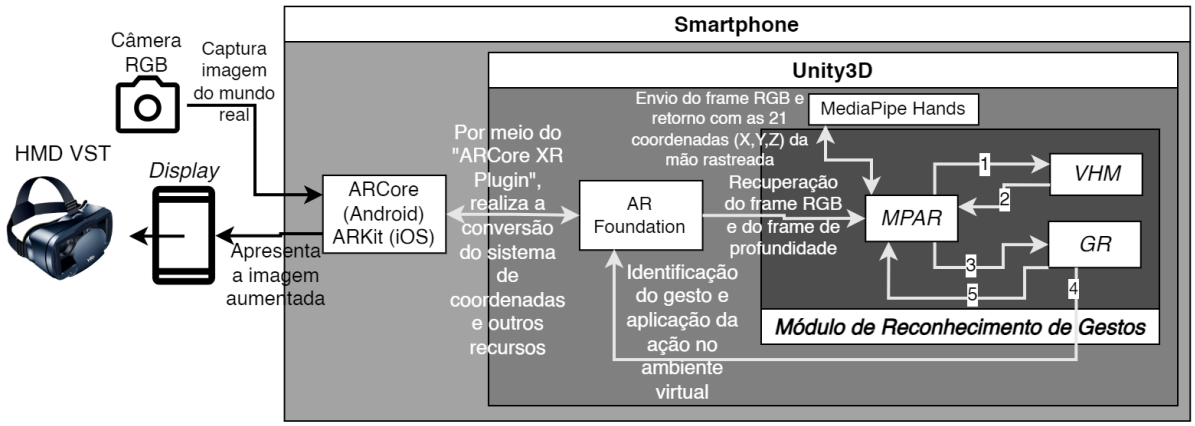
A arquitetura lógica do módulo pode ser vista na Figura 16 e a do sistema na Figura 17.

Figura 16 – Arquitetura lógica do módulo com abordagem baseada em MediaPipe Hands



Fonte: Elaborada pelo autor.

Figura 17 – Arquitetura lógica do sistema com abordagem baseada em MediaPipe Hands



Fonte: Elaborada pelo autor.

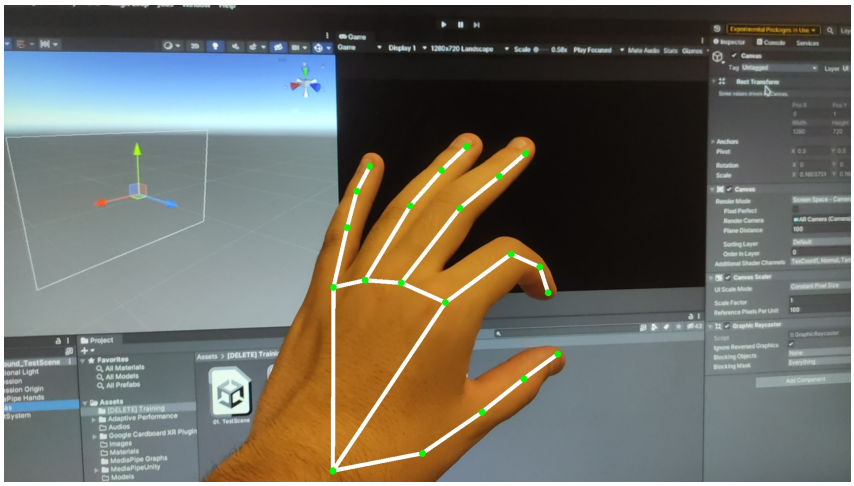
3.2.2.2 Desenvolvimento do Módulo

Munido da arquitetura planejada e da instalação das ferramentas principais, o primeiro passo para o desenvolvimento efetivo do módulo consistiu na instalação dos pacotes AR Foundation e do MediaPipe adaptado para o Unity3D.

Uma vez que o AR Foundation é um pacote oficial da própria Unity, a instalação foi realizada seguindo os passos expostos por Technologies (2022) na documentação oficial. Já com relação ao MediaPipe, a adaptação publicada por Nishida (2022) evidencia a relevância do sistema de destino para a chamada “construção nativa” do *plugin*. Em outras palavras, as funcionalidades do MediaPipe somente poderiam ser utilizadas após a sua compilação efetiva na máquina, tornando-a diferente para cada sistema

operacional. Desta forma, utilizou-se de um ambiente virtual completo para a instalação do *plugin*, por meio do chamado “Subsistema do Windows para Linux” (WSL). Com isso, pode testar e exportar essa adaptação para a Unity a fim de ser utilizado no projeto criado do módulo. O teste referido pode ser visto na Figura 18.

Figura 18 – Teste de rastreamento de mão com MediaPipe Hands adaptado para Unity



Fonte: Elaborada pelo autor.

Dado o exposto, iniciou-se efetivamente a construção do módulo com a criação de um objeto central para abranger três componentes funcionais, cada qual referente a uma parte estabelecida na arquitetura. Também, seguindo as diretrizes do AR Foundation, era necessário a utilização de dois objetos principais: AR Session, responsável por habilitar o sistema de RA; e o XR Origin, cuja função inclui desde o controle da câmera quanto ao dos rastreáveis (como planos).

Desta forma, o componente MPAR foi programado com base em três principais funções: configuração do MediaPipe Hands; processamento dos quadros obtidos da câmera de RA; e a obtenção de coordenadas das mãos rastreadas, se existentes. Juntamente a isso, o quadro relativo à profundidade era processado e interpretado, a partir das coordenadas obtidas pelo rastreamento de mão. Por fim, criou-se uma estrutura para armazenar as coordenadas: os valores da tripla ordenada (X, Y, Z) advindos do MediaPipe Hands; e o valor da profundidade (Z) da Depth API, em metros, e relativa ao par ordenado (X, Y) do rastreamento de mãos. Essa estrutura é utilizada para cada um dos 21 pontos de mão rastreados na cena.

Em seguida, construiu-se o VHM com base em três funções principais: recuperação das coordenadas do MPAR; construção das mãos virtuais, a partir de um modelo pré-definido com 21 esferas (simbolizando as coordenadas rastreadas); atualização das coordenadas das mãos virtuais criadas. Além disso, as mãos virtuais ficaram disponíveis para ser acessadas pelo futuro GR. Contudo, foi nesta etapa de

construção a qual evidenciou-se alguns dos principais problemas desta abordagem, uma vez que as mãos virtuais instanciadas não se tornavam visíveis em cena ou não tiveram suas posições alteradas corretamente de início. Mesmo assim, como o *software* não apresentava erros de compilação, o desenvolvimento foi prosseguido com o GR.

Por fim, desenvolveu-se o GR com base em duas funções principais: recebimento das mãos virtuais construídas pelo VHM; e identificação do gesto propriamente dita. Para identificar, utilizou-se um cálculo da distância relativa ao ponto do pulso (ponto 0) entre duas coordenadas de um gesto. Contudo, como o VHM apresentava incoerências, o GR permaneceu não funcionando corretamente. Vale ressaltar que os problemas encontrados nesta abordagem culminaram com a mudança de abordagem do projeto, uma vez que se tornou inviável a realização dos experimentos propostos na seção 4.1.

3.2.2.3 Dificuldades no Desenvolvimento

Durante o desenvolvimento desta abordagem, após a sua modelagem, alguns problemas puderam ser elencados, sendo cruciais para impossibilitar a avaliação do módulo para ambientes de RA. Por conta destes problemas, a abordagem do módulo foi inteiramente trocada, incluindo parte das ferramentas.

O primeiro contratempo, como citado anteriormente, ocorreu durante a fase de criação do VHM, especificamente com relação a construção da mão virtual na cena. Acredita-se que o problema esteja relacionado com o uso do AR Foundation, uma vez que a documentação não apresenta exemplos relacionados à inserção de objetos virtuais relativos à da câmera, mas sim, somente com o uso de marcadores fiduciais pré-cadastrados ou pela “localização e mapeamento simultâneos” (SLAM) de planos, uma vez que esse pacote se baseia diretamente nas funcionalidades ofertadas pelo ARCore.

O segundo problema crucial encontrado estava relacionado aos sistemas de coordenadas entre o *plugin* do MediaPipe e a Unity3D. Tendo em vista que a própria documentação do MediaPipe Hands não traz informações a respeito das coordenadas do objeto (mão) e da câmera para o rastreamento de mãos, tornou-se inviável a conversão entre os sistemas de coordenadas do *framework* para a Unity. Ademais, destaca-se o não rastreamento efetivo da profundidade da mão, apenas sendo relativo ao pulso e, desta forma, tornou essa abordagem inexequível para os experimentos propostos na seção 4.1.

3.3 Abordagem Final: Vuforia

Esta seção apresenta o processo de construção do módulo, por meio da abordagem do Vuforia, tendo em vista os problemas encontrados na abordagem do MediaPipe Hands. Em outras palavras, são descritos os materiais adicionais necessários, a modelagem e arquitetura do sistema e os detalhes no desenvolvimento.

3.3.1 Materiais Adicionais

Adicionalmente aos materiais gerais, tornou-se necessário a utilização dos seguintes materiais para essa abordagem:

Software

- Biblioteca de RA: Vuforia v10.12.3 para Unity3D.

Outros

- Dois marcadores distintos e impressos no tamanho 4,5 cm x 4,5 cm.

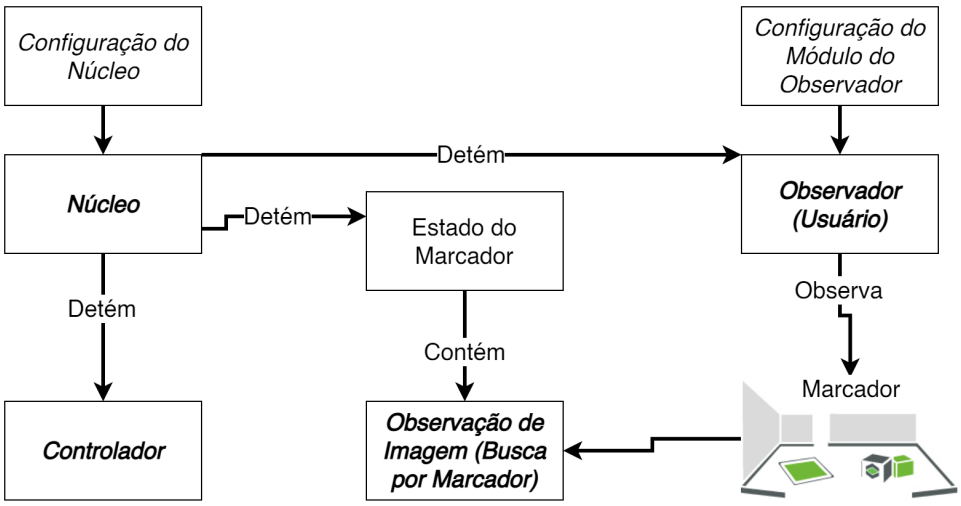
3.3.1.1 Vuforia

O Vuforia, de acordo com a PTC (2022), é um SDK de RA, cujo objetivo consiste em providenciar serviços para a criação de aplicações de RA baseadas em marcadores. Por conta de sua facilidade de uso, possibilidade de uso de HMDs VST e OST e estabilidade com relação ao rastreamento dos objetos na cena, esse SDK tem se consolidado como um dos mais utilizados na área.

Sua arquitetura pode ser vista na Figura 19 e seu funcionamento é baseado em três passos simples:

1. Previamente, os marcadores são cadastrados no sistema e, então, ocorre o cálculo dos chamados “pontos característicos”. Serão esses pontos que o Vuforia tentará encontrar no mundo real e, portanto, devem ser distinguíveis no ambiente.
2. Ao executar a aplicação, o Vuforia aciona o seu módulo de reconhecimento de imagem, o qual busca, a cada quadro recebido da câmera, os “pontos característicos” do marcador e realiza os devidos cálculos de rotação com os dados previamente salvos.
3. Caso um marcador seja encontrado, o gerenciador emite os cálculos de posição e um “sinal positivo” para que a Unity possa realizar a renderização dos elementos na cena. É importante ressaltar que, nesta abordagem, o marcador é utilizado como referência para a renderização dos objetos virtuais.

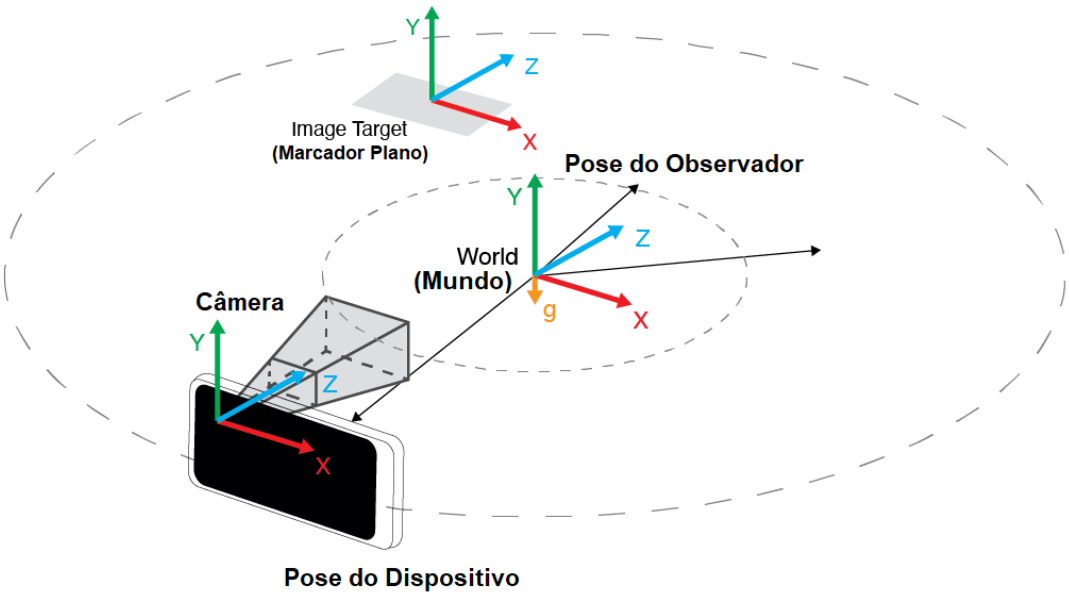
Figura 19 – Arquitetura interna de funcionamento do Vuforia



Fonte: Adaptada de PTC (2022).

Especificamente, na Figura 20 é possível ver a relação entre os sistemas de coordenadas de uma aplicação com Vuforia. Nota-se que o marcador rastreado possui seu próprio eixo de coordenadas, o qual permite a inferência de uma profundidade com relação aos objetos virtuais. Em outras palavras, o observador (inferido na figura por câmera), ao se aproximar de um objeto virtual, visualizará este com sua escala real, ao passo que, ao se afastar do mesmo objeto, visualizará este com sua escala diminuída.

Figura 20 – Sistemas de coordenadas utilizados pelo Vuforia na Unity3D

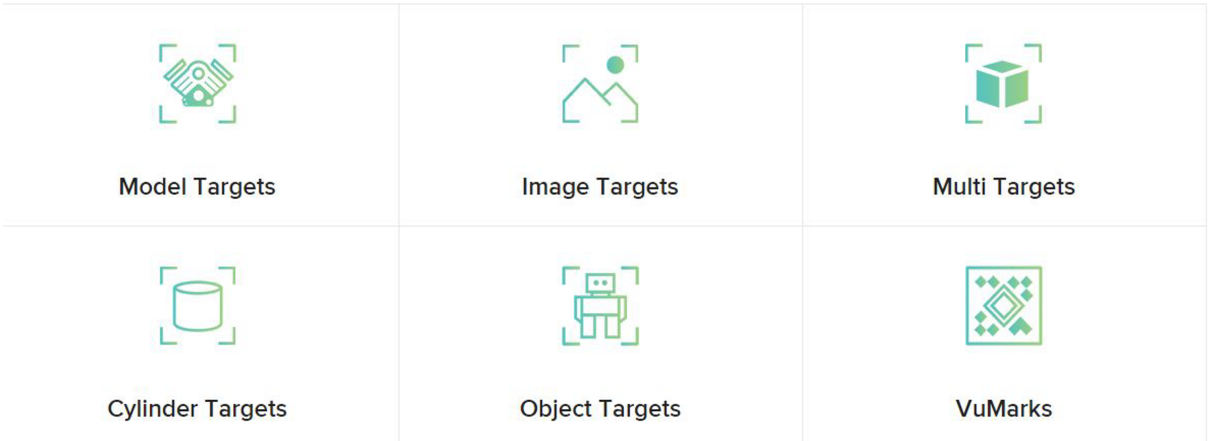


Fonte: Adaptada de PTC (2022).

Além disso, o Vuforia tem suporte para seis tipos de marcadores, cujo emblemas minimalistas podem ser vistos na Figura 21. São eles:

- Model Targets: permite com que os objetos físicos modelados digitalmente em 3D sejam reconhecidos e rastreados.
- Image Targets: um dos tipos mais simples, representam objetos planos, como imagens impressas, páginas de revistas, cartões comerciais.
- Multi Targets: representam objetos com superfícies planas de vários lados, sendo possível identificar várias imagens simultaneamente. Pode ser visto em embalagens de produtos.
- Cylinder Targets: neste tipo, os objetos de rastreamento são formas cilíndricas ou cônicas, como latas e garrafas.
- Object Targets: baseado através da varredura das superfícies de um objeto real, buscando cobrir todos os seus detalhes e de forma consistente.
- VuMarks: tipo específico de marcador plano semelhante a um código de barras.

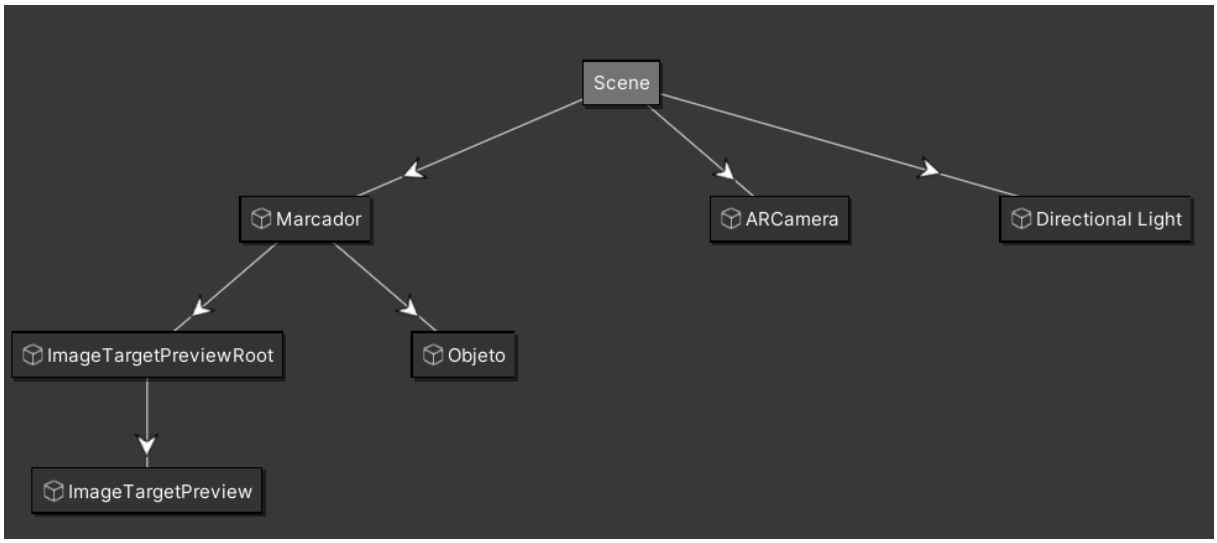
Figura 21 – Tipos de marcadores suportados pelo Vuforia, com ícones minimalistas



Fonte: PTC (2022).

Por conta da facilidade, esta abordagem se baseia na utilização de marcadores planos. O grafo de cena mínimo para a utilização do Vuforia com um marcador plano e um objeto virtual, na Unity, pode ser visto na Figura 22. Nota-se que o objeto virtual deve ser filho do marcador plano, uma vez que o Vuforia apenas indicará para a Unity a renderização de elementos que são dependentes de seus marcadores rastreados. Em outras palavras, os objetos virtuais devem ser dependentes dos marcadores cadastrados no Vuforia.

Figura 22 – Grafo de cena mínimo do Vuforia na Unity



Fonte: Elaborada pelo autor.

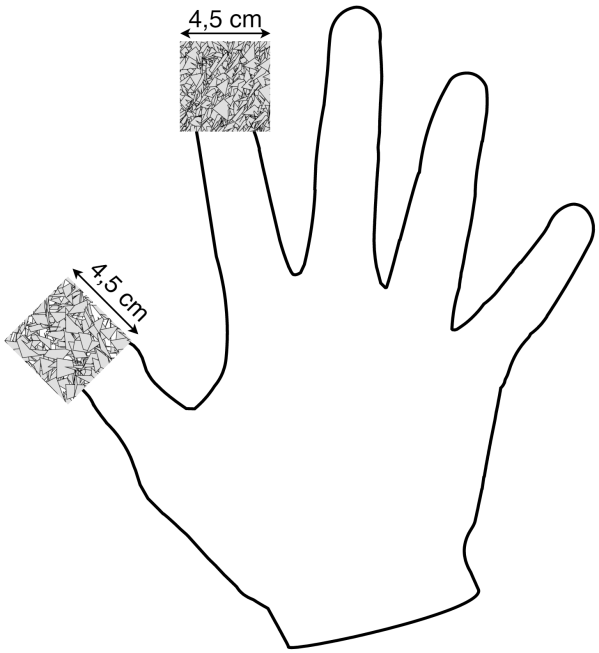
3.3.2 Métodos

Nesta seção, são apresentadas as modelagens e arquiteturas do módulo com a abordagem do Vuforia, bem como o desenvolvimento do mesmo.

3.3.2.1 Modelagem e Arquitetura do Sistema

Diante do exposto e dos materiais descritos, o módulo de reconhecimento de gestos nesta abordagem foi estabelecido levando em conta os dois marcadores fiduciais para os dedos polegar e indicador. O esquema de uso desses marcadores pode ser visto na Figura 23.

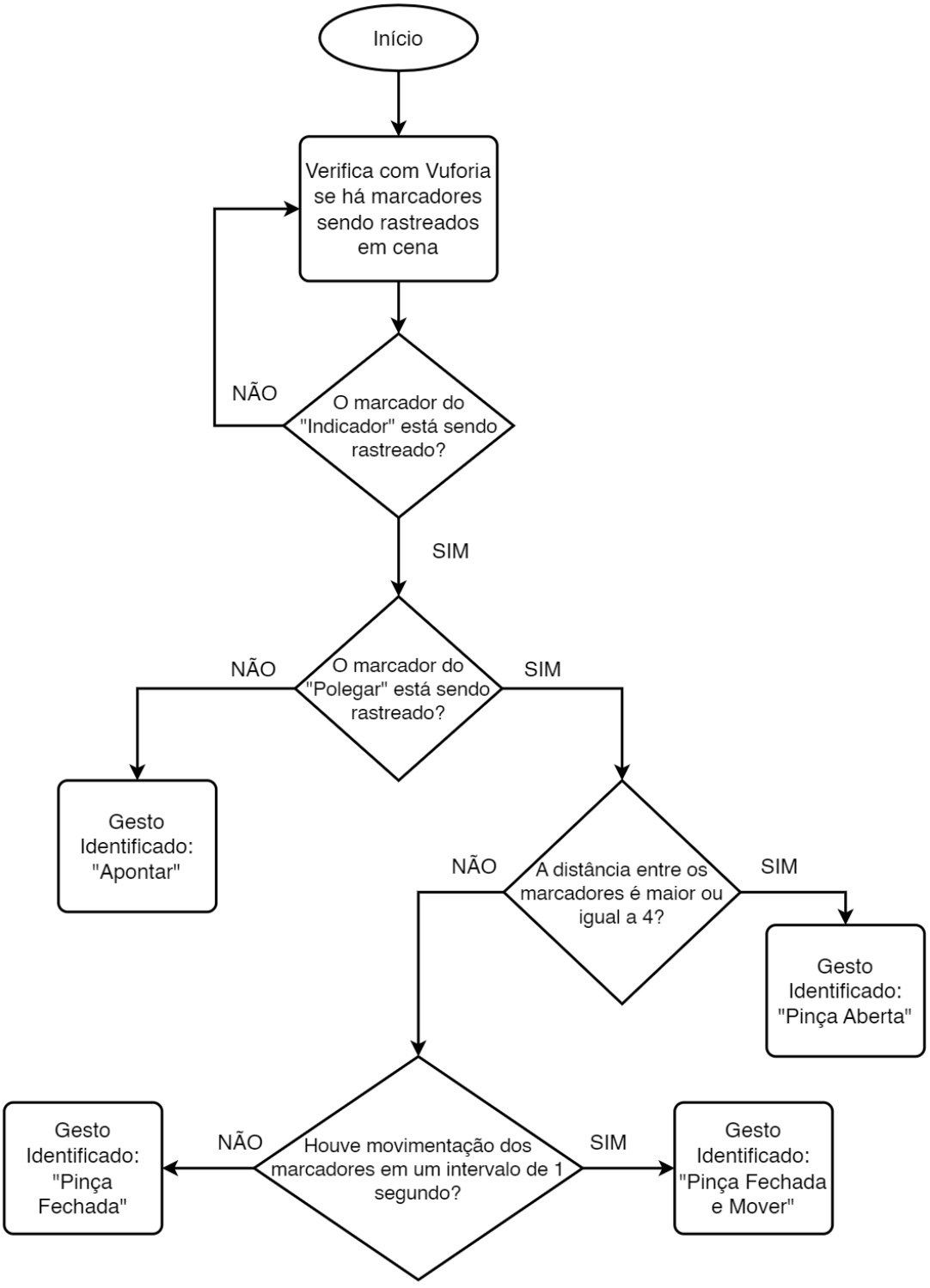
Figura 23 – Esquema de uso dos marcadores para identificação dos dedos



Fonte: Elaborada pelo autor.

A partir desta configuração, o módulo foi estruturado em duas principais partes: a classe de verificação dos marcadores dos dedos em cena e a classe de execução das ações dos gestos. A primeira pode ser descrita como a responsável por analisar e indicar, de forma contínua, o rastreamento dos marcadores dos dedos no mundo real, a partir da imagem obtida pela câmera. Para tanto, esse item realiza uma comunicação direta com o Vuforia, o qual emite um sinal para cada marcador rastreado na cena. Em caso positivo de rastreamento de pelo menos um dos marcadores dos dedos, é realizado a identificação de um suposto gesto por meio dos seguintes parâmetros: para o “apontar”, é necessário existir apenas o marcador do dedo indicador na cena; para os gestos de “pinça”, isto é, o “pinça fechada”, “pinça fechada e mover” e “pinça aberta”, são necessários o rastreio dos dois marcadores de dedos. Especificamente, os gestos de “pinça” são diferenciados por meio da distância dos marcadores, em que o gesto de “pinça fechada” somente irá existir se essa distância for menor que 4, ao passo que o “pinça aberta” existirá se a distância for maior ou igual a 4. Ademais, caso o gesto identificado seja a “pinça fechada”, é realizado um cálculo de movimento para verificar a existência do gesto “pinça fechada e mover”. Ou seja, caso haja uma movimentação, dentro do intervalo de um segundo, com relação aos marcadores, então o gesto “pinça fechada e mover” é ativado. O fluxograma de funcionamento deste item pode ser visto na Figura 24.

Figura 24 – Fluxograma de funcionamento do identificador de gestos



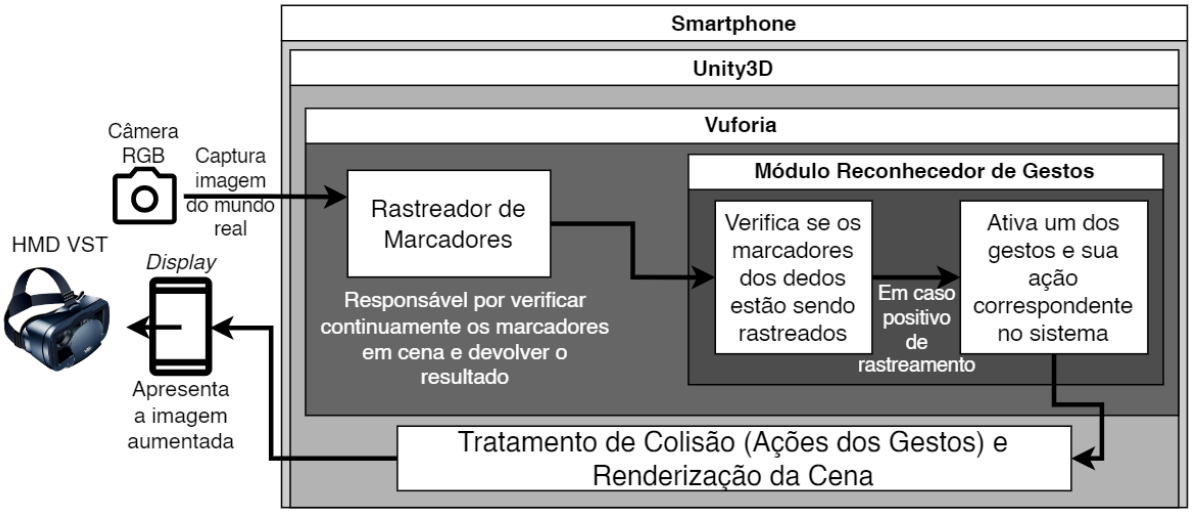
Fonte: Elaborada pelo autor.

Tendo realizado esse processo de identificação, a informação obtida é comunicada para a segunda classe, responsável por acionar a ação contínua associada a um gesto identificado em cena. Em outras palavras, isso simboliza que, caso haja, de fato, a existência de um gesto, por meio da indicação da primeira classe, a segunda classe

realiza a execução da ação, de forma constante e ininterrupta, do gesto associado, somente cessando após a perda do marcador ou da não identificação de um gesto pela primeira classe. Essas ações associadas podem ser acessadas facilmente por outros objetos, com a finalidade de facilitar a comunicação entre as partes do sistema.

Em síntese, a arquitetura lógica do sistema modelado pode ser vista na Figura 25.

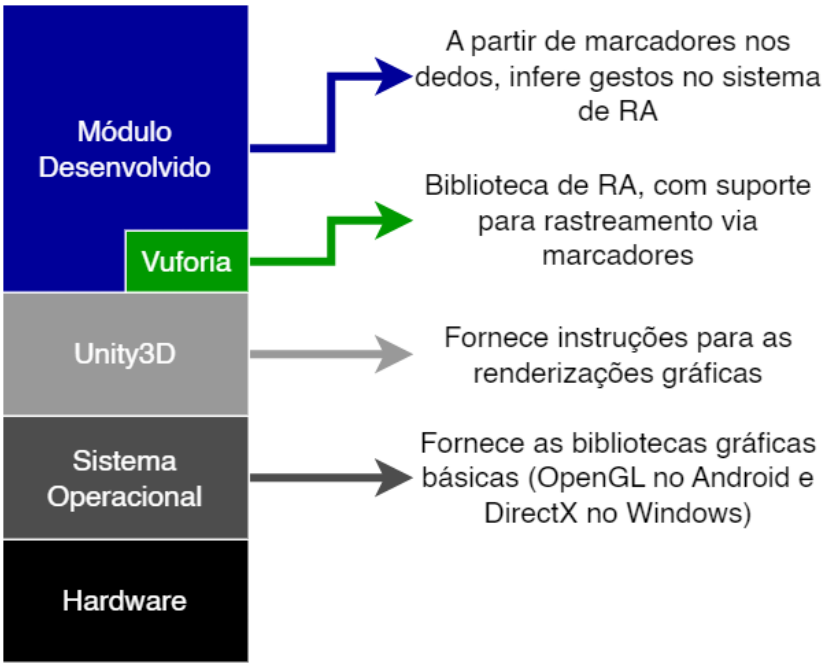
Figura 25 – Arquitetura lógica do sistema com a abordagem Vuforia



Fonte: Elaborada pelo autor.

Além disso, pode-se ilustrar a estrutura hierárquica de *software* por meio da Figura 26. Nessa estrutura, nota-se que nos níveis mais baixos, representados pelo *hardware* e sistema operacional, ocorre o chamado “processamento bruto” das informações. Em outras palavras, são fornecidas estruturas gráficas básicas para o tratamento bruto da renderização dos objetos virtuais, o que não é trivial no desenvolvimento por exigirem uma série de instruções para o devido uso. Por isso, a ferramenta Unity aparece logo acima, de tal maneira a oferecer uma interface para essas estruturas básicas, além de ofertar uma organização e configuração do ambiente virtual por meio dos grafos de cena. Ademais, a Unity facilita na criação e integração de diferentes bibliotecas de RA, como o Vuforia, localizado acima da Unity. Por se tratar de uma biblioteca de RA, o Vuforia oferta uma série de funcionalidades para o registro e rastreamento de objetos na cena, os quais são indispensáveis para a existência da RA. Diante disso, o módulo desenvolvido nesta abordagem se localiza na camada mais acima possível, uma vez que se utiliza dos serviços do Vuforia e foi feito, especificamente, para uso na Unity.

Figura 26 – Estrutura hierárquica de *software* do módulo

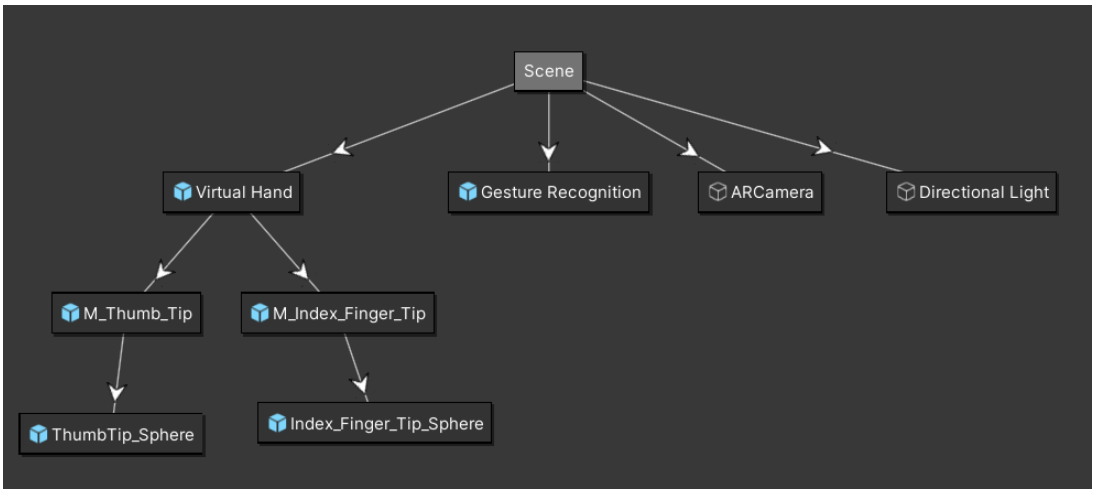


Fonte: Elaborada pelo autor.

3.3.2.2 Desenvolvimento do Módulo

Inicialmente, construiu-se um objeto modelo (*prefab*) que controla cada um dos marcadores dos dedos, com o intuito de facilitar a transmissão de informações com os outros objetos do sistema, como o módulo. Além disso, associou-se duas esferas de cor verde e com tamanho mínimo para cobrir cada marcador na cena renderizada, de tal forma a melhorar a imersão do usuário no sistema, possibilitar a existência de colisão dos dedos com os objetos virtuais e realizar os cálculos de distância e movimento previstos na estruturação do módulo. O grafo de cena representando a cena mínima de objetos do módulo pode ser visto na Figura 27. Destaca-se que o objeto bruto do módulo é representado pelo “Gesture Recognition” (Reconhecimento de Gestos), ao passo que a “mão virtual” (Virtual Hand) realiza o controle direto dos marcadores “M_Thumb_Tip” (dedo polegar) e “M_Index_Finger_Tip” (dedo indicador), com cada qual possuindo uma esfera como filho direto.

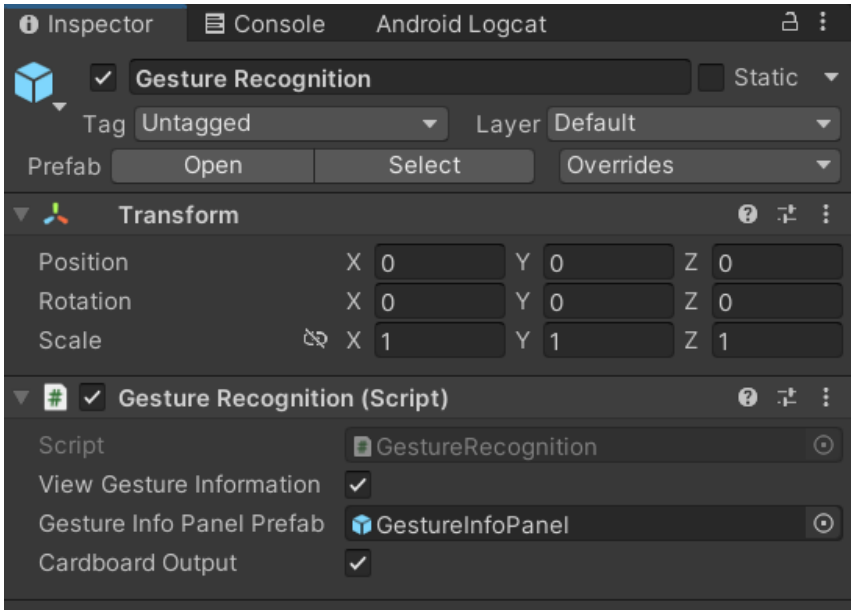
Figura 27 – Grafo de cena mínimo para utilização do módulo



Fonte: Elaborada pelo autor.

Havendo essas definições, criou-se um componente do tipo *script* para o objeto representativo do módulo. Esse componente abrange as duas funcionalidades previstas na arquitetura, isto é, a identificação dos gestos por meio do rastreamento dos marcadores de dedos e a inferência de uma ação associada ao gesto. Ademais, implementou-se uma interface em texto para informar, em tempo real, a respeito do reconhecimento de gestos. O objeto final do módulo, isto é, aquele que deve ser importado na aplicação para usufruir dos serviços de reconhecimento de gestos, obviamente com o objeto de mão virtual corretamente configurado, pode ser visto na Figura 28.

Figura 28 – Objeto final representativo do módulo



Fonte: Elaborada pelo autor.

4 Experimentação e Análise dos Resultados

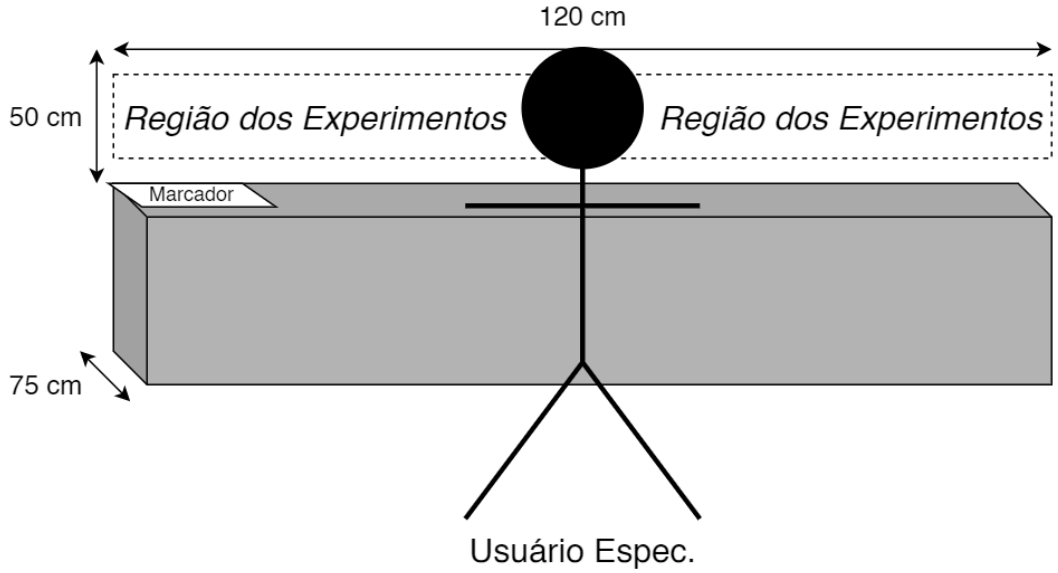
Neste capítulo são apresentados os experimentos realizados para a medição da acurácia do módulo bem como a análise dos resultados obtidos.

4.1 Experimentos

Com o propósito de avaliar a acurácia do módulo para os ambientes de RA HMDs VST baseados em *smartphones*, propôs-se a utilização de medidas quantitativas fundamentada na captura das mãos do usuário em determinadas ações para diferentes cenários de RA.

Desta forma, optou-se por seguir uma metodologia semelhante a utilizada por Moser e Swan (2016), cujos testes foram realizados por somente um usuário especializado de tal forma que fosse possível avaliar efetivamente a interação no espaço de RA. Além disso, adaptou-se a metodologia empregada com a de Batistão (2020) e Piumsomboon et al. (2013), de tal forma que o usuário especializado realizaria as interações em um cenário de RA que simula uma tarefa. Foram formulados dois experimentos, cada qual sendo específico para as ações dos gestos propostos neste trabalho. Para cada experimento, criou-se dois cenários de teste progressivos, cada qual devendo ser executado em três ciclos. Um ciclo envolve a execução do cenário em três vezes consecutivas. Além disso, propôs-se um tempo de 5 segundos entre cada ciclo de cada cenário e um tempo de 10 segundos para a troca de cenário. Ressalta-se, também, que os cenários cobriram uma região de interação com 120 cm x 50 cm x 75 cm (Comprimento x Altura x Profundidade), conforme é mostrado na Figura 29. Além disso, utilizou-se um marcador impresso no tamanho 20 cm x 29 cm (Altura x Largura, na forma horizontal) colocado no canto central esquerdo com intuito de servir como centro de referência no Vuforia para os experimentos.

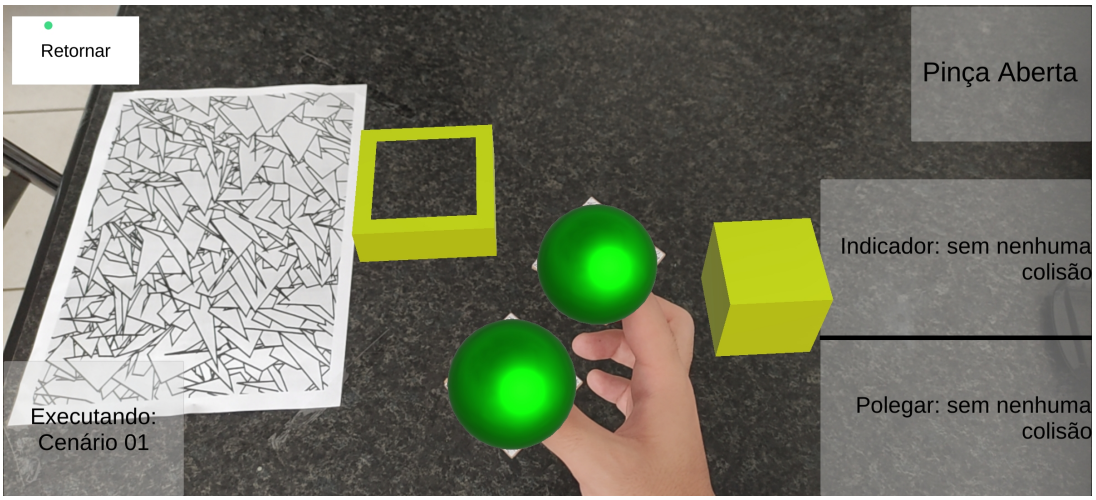
Figura 29 – Região de interação dos experimentos



Fonte: Elaborada pelo autor.

O primeiro experimento foi específico para os gestos relacionados a seleção direta com os objetos virtuais, ou seja, testou-se a acurácia de colisão dos gestos de “pinça fechada”, “pinça fechada e mover” e “pinça aberta”. Diante disso, a tarefa exigida pelos dois cenários de teste consiste em selecionar e arrastar objetos virtuais, colocando-os num alvo correspondente. No primeiro cenário, definiu-se apenas um objeto afastado em 15 centímetros de seu alvo, munindo da aleatoriedade de forma e cor. Já no segundo cenário, definiu-se a quantidade de dois objetos, cada qual afastado em 15 centímetros de seu alvo, novamente com a escolha aleatória de forma e cor. Ressalta-se que cada alvo é restrito a um objeto definido, de tal forma que o usuário deve encaixar os objetos em seus respectivos alvos. Para tanto, o usuário se utilizou de três gestos do sistema: “pinça fechada”, para selecionar e agarrar um objeto; “pinça fechada e mover”, para mover o objeto para a posição do alvo; e “pinça aberta”, para liberar o objeto no alvo. É possível visualizar um exemplo do primeiro experimento na Figura 30.

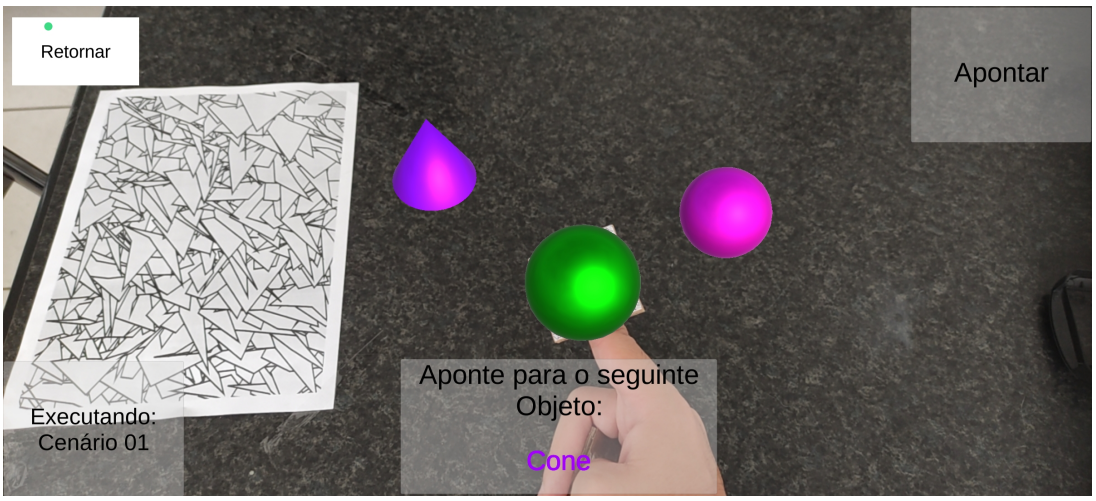
Figura 30 – Exemplo do experimento 01



Fonte: Elaborada pelo autor.

Já o segundo experimento foi específico para o gesto de “apontar”, relacionado a seleção indireta, isto é, sem colisão, com os objetos virtuais. Neste contexto, o sistema gera uma certa quantidade de objetos aleatoriamente dentro da região definida, cada qual com sua forma e cor, e solicita a seleção de um específico ao usuário. O usuário, diante disso, deve utilizar o gesto “apontar”, o qual cria um raio de profundidade a partir da esfera presente na ponta do dedo indicador, e selecionar o objeto solicitado. Diante disso, definiu-se a quantidade de dois objetos para o primeiro cenário e de cinco objetos para o segundo cenário. É possível visualizar um exemplo do segundo experimento na Figura 31.

Figura 31 – Exemplo do experimento 02



Fonte: Elaborada pelo autor.

Por fim, as métricas descritas no Quadro 1 foram utilizadas para cada cenário

de cada experimento, de tal modo que os dados obtidos pudessem ser comparados de modo equivalente.

Quadro 1 – Métricas para os Experimentos

Medida	Descrição
Tempo Mínimo de Reconhecimento	Tempo decorrido do sistema, em segundos, para identificar qualquer gesto inicial
Tempo de Teste	Tempo decorrido, em segundos, para terminar um teste de um ciclo
Erro de Distância	Distância, em centímetros, entre centro do objeto e: respectivo alvo ou centro do dedo indicador na seleção correta
Erro de Alvo	Distância, em centímetros, entre centro do objeto e: alvo incorreto ou centro do dedo indicador na seleção incorreta
Contador de Cada Gesto	Quantidade inteira total para cada gesto realizado pelo usuário

Fonte: Elaborado pelo autor.

4.2 Resultados e Discussões

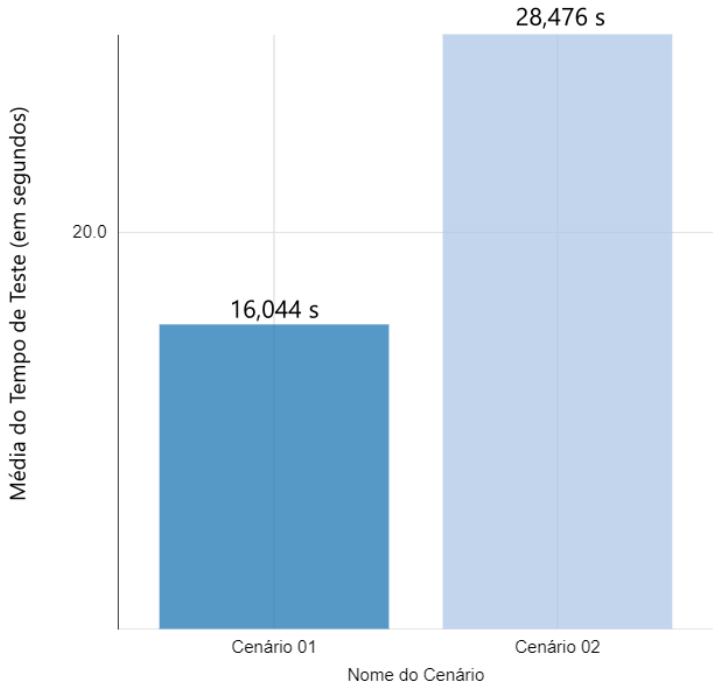
Tendo em vista a execução de três testes por ciclo e três ciclos por cenário, para cada um dos dois experimentos realizados e, ainda, levando em conta os diversos erros de alvos obtidos das seleções incorretas de objetos virtuais no segundo experimento, obteve-se o total de 3820 medidas considerando os dois experimentos juntos. Especificamente, foram 27 medidas para o experimento 1 e 3793 para o experimento 2, por conta das seleções incorretas de objetos.

Diante disso, as medidas obtidas foram organizadas e tratadas com a finalidade de obtenção das respectivas médias. Essas médias denotam o ponto de equilíbrio do conjunto de dados dos experimentos, sendo considerada uma representação significativa do funcionamento do módulo nos ambientes propostos. Tendo em vista que cada experimento proposto detinha a finalidade de avaliar uma parte do módulo, os resultados serão apresentados e discutidos com base nos cenários de cada um dos experimentos.

4.2.1 Experimento 01

Inicialmente, o gráfico da média do tempo de teste presente na Figura 32 evidencia que o usuário demorou cerca de 16 segundos para completar o primeiro cenário, ao passo que demorou cerca de 28 segundos para completar o segundo cenário. A justificativa para isso se deve pela complexidade dos cenários, isto é, enquanto o primeiro possui apenas um objeto para ser encaixado no seu respectivo alvo, o segundo possui dois objetos distintos e, naturalmente, deve gastar mais tempo para ser realizado do que o seu antecessor.

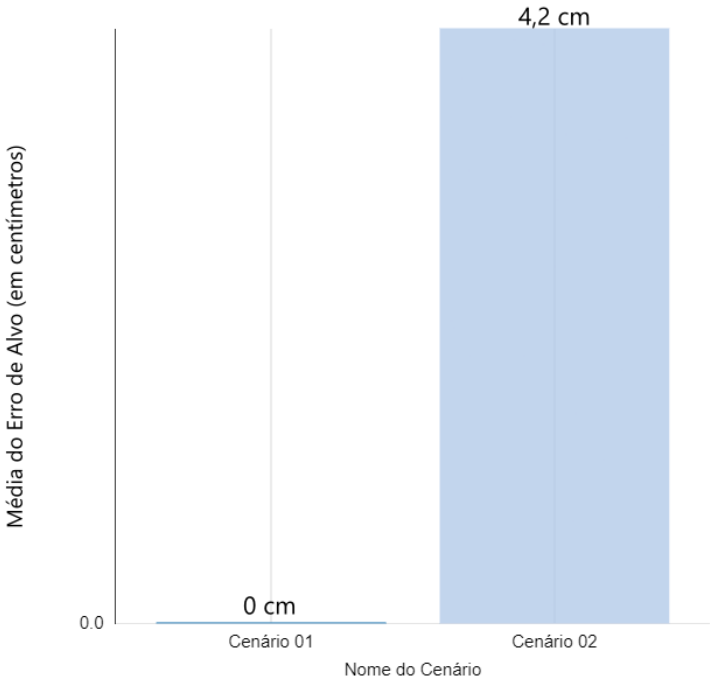
Figura 32 – Gráfico da média do tempo de teste do experimento 01



Fonte: Elaborada pelo autor.

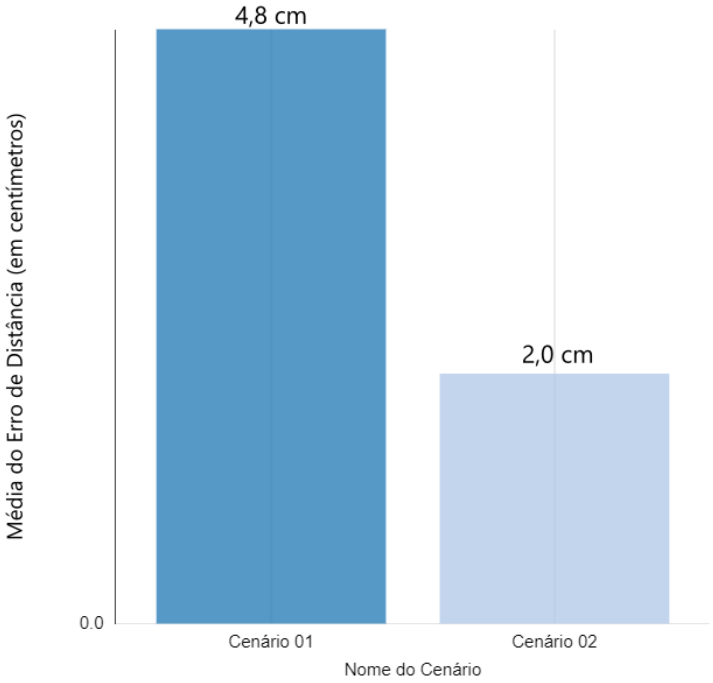
Em meio ao exposto, é possível justificar-se igualmente tanto para a média do erro de alvo dos objetos ser maior no segundo cenário do que no primeiro, quanto para a média do erro de distância ser maior no primeiro cenário do que no segundo. Respectivamente, essas médias podem ser vistas na Figura 33 e na Figura 34. Ressalta-se a importância dessas métricas encontradas, uma vez que simbolizam a efetividade da ação dos gestos de “pinça” no ambiente. Em outras palavras, o valor médio de erro de distância de 4,8 cm para o primeiro cenário e o valor médio de erro de alvo de 4,2 cm para o segundo cenário, os quais simbolizam os maiores com relação as métricas de erros de distância dos objetivos, ainda sim, são relativamente baixas. Em poucas palavras, isso indica que o módulo desenvolvido está atendendo a realização de ações no espaço peripessoal, ainda que não haja uma grande precisão.

Figura 33 – Gráfico da média do erro de alvo do experimento 01



Fonte: Elaborada pelo autor.

Figura 34 – Gráfico da média do erro de distância do experimento 01



Fonte: Elaborada pelo autor.

Além disso, destaca-se que o sistema possibilitou um tempo bastante semelhante, quase idêntico, no que diz respeito a identificação de qualquer primeiro gesto

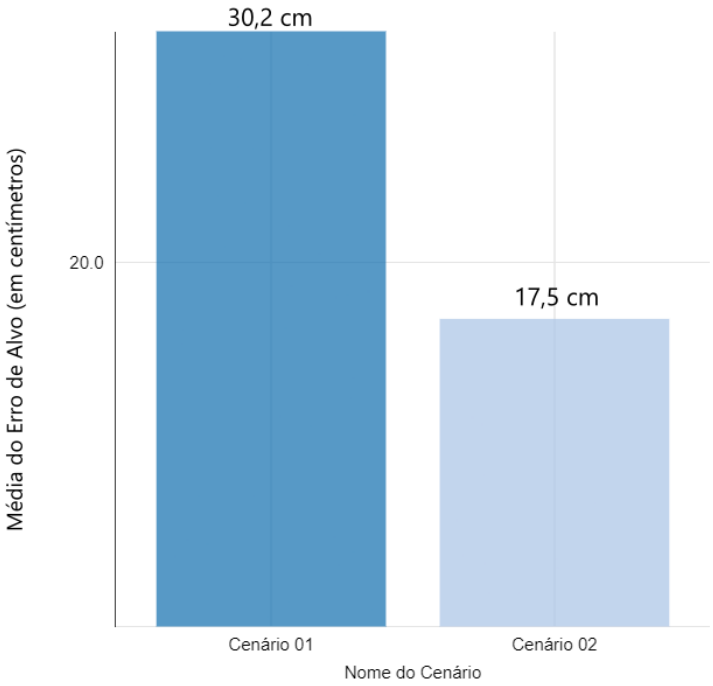
na cena. O tempo médio de reconhecimento foi de 2 segundos, ainda que isso possa variar com base na iluminação do local, da qualidade da câmera de RA ou até mesmo do tamanho e qualidade dos marcadores fiduciais.

Por fim, com relação a quantidade de cada gesto identificado durante os experimentos dos cenários, é possível destacar uma certa diferença entre os valores dos dois cenários: para o “apontar”, foi encontrado o total de 2 vezes no primeiro cenário e 11 vezes no segundo; para a “pinça fechada”, novamente o valor de 2 vezes para o primeiro cenário, mas 16 vezes para o segundo; para a “pinça fechada e mover” foram o total de 11 identificações no primeiro cenário e 18 no segundo; e para a “pinça aberta”, o primeiro cenário identificou 11 vezes enquanto que o segundo cenário identificou 20 vezes. Uma das justificativas plausíveis para essas diferenças, novamente, se baseia na complexidade dos dois sistemas, uma vez que é exigido a realização de mais ações no segundo cenário do que no primeiro.

4.2.2 Experimento 02

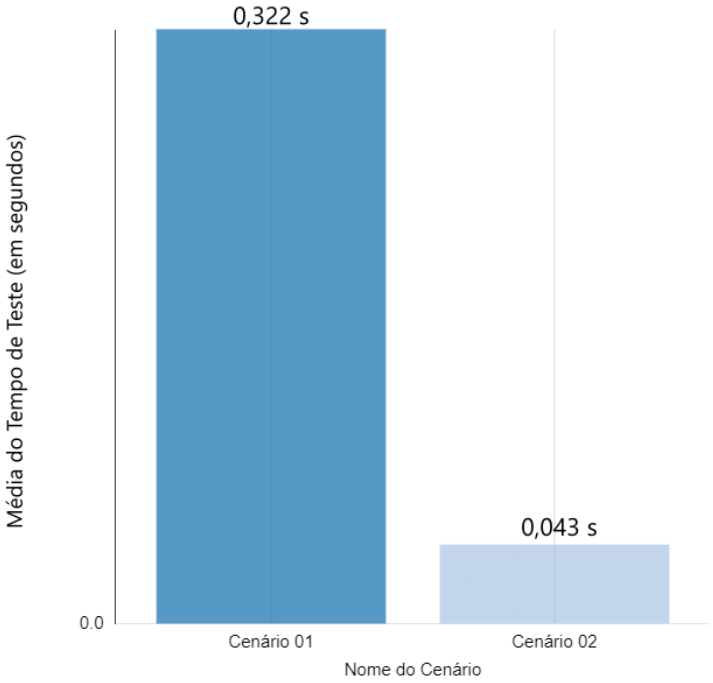
Tendo em vista que neste experimento o intuito era avaliar o gesto “apontar” num cenário de escolha de objetos virtuais, foi possível coletar e evidenciar a presença de inúmeras indicações incorretas de objetos. Em outras palavras, é possível visualizar na Figura 35 que o valor médio de erro de alvo para o primeiro cenário, o qual possuía apenas três objetos para indicação, foi em torno de 30 cm, considerado um valor de erro alto na indicação. Contudo, a média de erro de alvo no segundo cenário, considerado mais complexo que o primeiro, foi em torno de 17 cm. Uma justificativa deste cenário consiste em problemas de geração dos objetos nos cenários, uma vez que se propôs a geração aleatória dos objetos virtuais e este fato pode ter ocasionado estas métricas alteradas. Isso também pode ter acarretado na alteração dos valores médios do tempo de teste dos cenários, vistos na Figura 36. Nota-se que esses valores são próximos de zero, o que é logicamente impossível de ser realizado.

Figura 35 – Gráfico da média do erro de alvo do experimento 02



Fonte: Elaborada pelo autor.

Figura 36 – Gráfico da média do tempo de teste do experimento 02

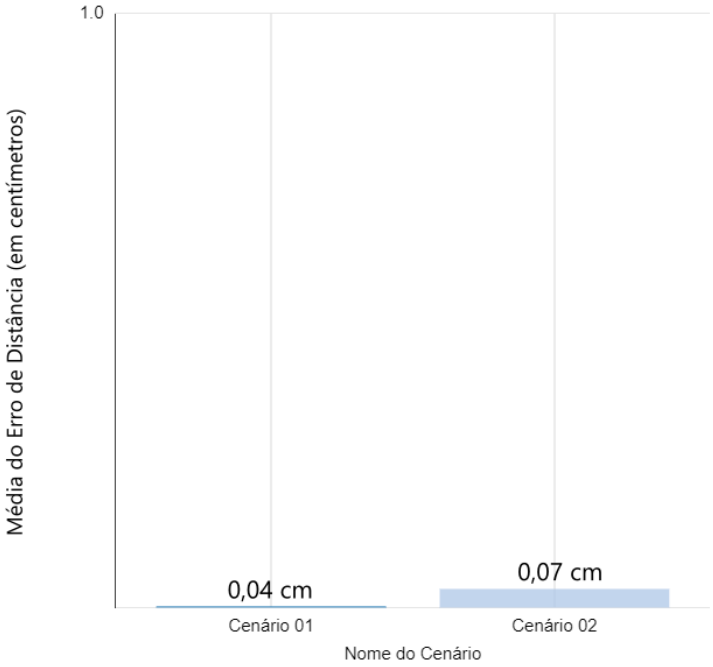


Fonte: Elaborada pelo autor.

Por outro lado, a média do erro de distância com valores próximos de zero para ambos os cenários, apresentada na Figura 37, trouxe à tona a inferência de uma

possível precisão do gesto “apontar” quando utilizado devidamente com os objetos solicitados pelo sistema. Além disso, evidencia-se, assim como no experimento 1, que o sistema possibilitou um tempo quase idêntico para a identificação de qualquer primeiro gesto na cena. Novamente, o tempo médio de reconhecimento foi de 2 segundos.

Figura 37 – Gráfico da média do erro de distância do experimento 02



Fonte: Elaborada pelo autor.

Por fim, com relação a quantidade do gesto “apontar” identificado durante os experimentos dos cenários, obteve-se o valor de 1469 para o primeiro cenário e 2324 para o segundo cenário. Como este cenário não exigia as ações ofertadas pelos gestos de “pinça”, desta forma, eles não foram contabilizados. Com relação aos valores elevados, novamente, podem ser justificados por problemas na geração aleatória dos objetos virtuais. Em curto prazo, uma alternativa a isso consiste na predefinição das posições iniciais dos elementos virtuais criados, o que poderia ocasionar estabilidade aos resultados obtidos. Contudo, ainda que alguns deles tenham sido inferidos incorretamente, o módulo demonstrou-se estável e confiável para a seleção de objetos específicos exigidos pelo sistema, conforme é mostrado pelas métricas do erro de distância.

5 Considerações Finais

O objetivo principal desse trabalho foi estruturar, implementar e avaliar um módulo de reconhecimento de gestos para interações em ambientes de RA restritos ao espaço peripessoal do usuário. Especificamente, os ambientes foram direcionados aos HMDs VST baseados em *smartphone*, uma vez que representam uma solução em baixo custo e com alto potencial de aplicação na sociedade. Diante disso, optou-se inicialmente pelo uso dos serviços de rastreamento de mão oferecidos pelo MediaPipe Hands, dado que são otimizados para aparelhos móveis e destinados para aplicações de RA. Contudo, as dificuldades encontradas durante o desenvolvimento do módulo por esta abordagem tornaram necessário a reestruturação do sistema por completo, desta vez munido pelo *software* consolidado Vuforia. Ressalta-se, ainda, a escolha da interação baseada em gestos pautada pela justificativa de naturalidade do usuário, isto é, trata-se de um método intuitivo e que não requer muito treinamento para ser utilizado. Também, levantou-se alguns dos gestos mínimos de mão para uma aplicação de RA, com base em sua função: apontar (seleção de objetos à distância); pinça fechada (seleção de objetos com colisão); pinça fechada e mover (transladar objetos); pinça aberta (liberar ação sobre objetos). Além disso, foram revistos conceitos de RA, técnicas de interação em ambientes de RA, gestos de mão como ações em ambientes virtuais, sistemas de coordenadas dos ambientes virtuais, modelagem de *software* baseada em módulos, funcionamento de tecnologias vigentes de RA (como AR Foundation, ARCore e Vuforia), arquitetura dos HMDs VST, dentre outras temáticas.

Diante desta nova abordagem com o Vuforia, isto é, com o uso de três marcadores de identificação, sendo um para o dedo indicador, um para o dedo polegar e um para os cenários dos experimentos, foi possível estruturar, implementar e avaliar um módulo, em *software*, capaz de reconhecer quatro gestos mínimos em aplicações de RA para HMDs VST baseadas em *smartphone* e, sobretudo, sem utilizar qualquer aparelho externo ao celular. Durante o processo, criou-se um ambiente experimental controlado capaz de realizar algumas interações naturais baseadas em gestos de mão para diferentes tarefas. Ainda que houveram problemas relacionados a profundidade e oclusão, os resultados do módulo com esta abordagem indicaram o possível emprego dos HMD VST para aplicações de RA, baseadas em gestos de mão, que não exijam alta precisão, como nas áreas da Educação, Jogos Digitais, Marketing, dentre outras.

5.1 Trabalhos Futuros

Considerando uma futura continuidade do trabalho, os seguintes aspectos podem ser explorados:

- Utilização de uma abordagem direta (ARCore) de construção de ambientes em RA;
- Expansão dos cenários experimentais propostos;
- Utilização conjunta do MediaPipe Hands e Vuforia;
- Investigação do uso com HMDs OST baseados em *smartphone*;
- Integração de mais gestos e mais ações no sistema;
- Realização de uma avaliação qualitativa;
- Aumento da quantidade de mãos rastreadas simultaneamente;
- Utilização em sistemas colaborativos.

Referências

AL-HAMMADI, M.; MUHAMMAD, G.; ABDUL, W.; ALSULAIMAN, M.; BENCHERIF, M. A.; MEKHTICHE, M. A. Hand Gesture Recognition for Sign Language Using 3DCNN. *IEEE Access*, v. 8, p. 79491–79509, 2020. ISSN 2169-3536.

ALECRIM, E. *Xiaomi Poco X3: o intermediário gamer*. 2020. Disponível em: https://tecnoblog.net/testamos/xiaomi-poco-x3-nfc-review/. Acesso em: 03 jan. 2023.

ALIPRANTIS, J.; KONSTANTAKIS, M.; NIKOPOULOU, R.; MYLONAS, P.; CARIDAKIS, G. Natural Interaction in Augmented Reality context. In: . [s.n.], 2019. Disponível em: http://www.image.ece.ntua.gr/papers/959.pdf. Acesso em: 03 jan. 2023.

AMEUR, S.; KHALIFA, A. B.; BOUHLEL, M. S. Hand-Gesture-Based Touchless Exploration of Medical Images with Leap Motion Controller. In: *2020 17th International Multi-Conference on Systems, Signals Devices (SSD)*. [S.l.: s.n.], 2020. p. 6–11.

ARENA, F.; COLLOTTA, M.; PAU, G.; TERMINE, F. An Overview of Augmented Reality. *Computers*, v. 11, n. 2, 2022. ISSN 2073-431X. Disponível em: https://www.mdpi.com/2073-431X/11/2/28. Acesso em: 25 maio 2022.

ARNALDI, B.; GUITTON, P.; MOREAU, G. *Virtual Reality and Augmented Reality: Myths and Realities*. 1. ed. [S.l.]: Wiley-IEEE Press, 2018. ISBN 1-78630-105-9.

AUKSTAKALNIS, S. *Practical Augmented Reality*. Boston, MA: Addison-Wesley Educational, 2017.

AZUMA, R. T. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, v. 6, n. 4, p. 355–385, ago. 1997. Disponível em: https://doi.org/10.1162/pres.1997.6.4.355. Acesso em: 03 jan. 2023.

BARFIELD, W. Wearable Computers and Augmented Reality: Musings and Future Directions. In: BARFIELD, W. (Ed.). *Fundamentals of wearable computers and augmented reality*. 2. ed. London, England: CRC Press, 2017. p. 3–11.

BATISTãO, L. V. *Percepção e Interação no Espaço Peripessoal de Realidade Aumentada Utilizando HMDs Video See-Through Baseados em Smartphone: Um Estudo de Caso*. Tese (Trabalho de Conclusão de Curso) — Faculdade de Ciências, Universidade Estadual Paulista "Júlio de Mesquita Filho", Bauru, 2020. Disponível em: https://dco-unesp-bauru.github.io/tcc-bcc-2020-2/LuccaVB/thesis-LuccaVB.pdf. Acesso em: 03 jan. 2023.

BILLINGHURST, M.; CLARK, A.; LEE, G. A survey of augmented reality. v. 8, n. 2, p. 73–272, 2015. ISSN 1551-3955, 1551-3963. Disponível em: http://www.nowpublishers.com/article/Details/HCI-049. Acesso em: 03 jan. 2023.

BUFACCHI, R. J.; IANNETTI, G. D. An Action Field Theory of Peripersonal Space. *Trends in Cognitive Sciences*, v. 22, n. 12, p. 1076–1090, 2018. ISSN 1364-6613. Disponível em: https://www.sciencedirect.com/science/article/pii/S1364661318302225. Acesso em: 03 jan. 2023.

GHAZWANI, Y.; SMITH, S. Interaction in Augmented Reality: Challenges to Enhance User Experience. In: *Proceedings of the 2020 4th International Conference on Virtual and Augmented Reality Simulations*. New York, NY, USA: Association for Computing Machinery, 2020. (ICVARS 2020), p. 39–44. ISBN 978-1-4503-7694-5. Disponível em: https://doi.org/10.1145/3385378.3385384. Acesso em: 03 jan. 2023.

GOOGLE. *MediaPipe Documentation*. 2020. Disponível em: https://google.github.io/mediapipe/. Acesso em: 03 jan. 2023.

GOOGLE. *ARCore Documentation*. 2022. Disponível em: https://developers.google.com/ar/develop. Acesso em: 03 jan. 2023.

HOUNSELL, M. da S.; TORI, R.; KIRNER, C. Realidade Aumentada. In: TORI, R.; HOUNSELL, M. d. S. (Ed.). *Introdução a Realidade Virtual e Aumentada*. 3. ed. Porto Alegre: Editora SBC, 2020. p. 30–59. ISBN 978-65-87003-54-2. Disponível em: https://doi.org/10.5753/sbc.6654.2. Acesso em: 03 jan. 2023.

HUNLEY, S. B.; LOURENCO, S. F. What is peripersonal space? An examination of unresolved empirical issues and emerging findings. *WIREs Cognitive Science*, v. 9, n. 6, p. e1472, 2018. Disponível em: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcs.1472. Acesso em: 03 jan. 2023.

KHURSHID, A.; GRUNITZKI, R.; LEYVA, R. G. E.; MARINHO, F.; ORLANDO, B. M. M. S. Hand Gesture Recognition for User Interaction in Augmented Reality (AR) Experience. In: CHEN, J. Y. C.; FRAGOMENI, G. (Ed.). *Virtual, Augmented and Mixed Reality: Design and Development*. Cham: Springer International Publishing, 2022. p. 306–316. ISBN 978-3-031-05939-1. Disponível em: https://link.springer.com/chapter/10.1007/978-3-031-05939-1_20. Acesso em: 03 jan. 2023.

KIYOKAWA, K. Head-mounted display technologies for augmented reality. In: BARFIELD, W. (Ed.). *Fundamentals of Wearable Computers and Augmented Reality*. 2. ed. [S.l.]: CRC Press, 2015. p. 59–84. ISBN 978-1-138-74931-3.

LUGARESI, C.; TANG, J.; NASH, H.; MCCLANAHAN, C.; UBOWEJA, E.; HAYS, M.; ZHANG, F.; CHANG, C.-L.; YONG, M. G.; LEE, J.; CHANG, W.-T.; HUA, W.; GEORG, M.; GRUNDMANN, M. *MediaPipe: A Framework for Building Perception Pipelines*. arXiv, 2019. Disponível em: http://arxiv.org/abs/1906.08172. Acesso em: 03 jan. 2023.

MICROSOFT. *HoloLens 2—Pricing and Options | Microsoft HoloLens*. 2022. Disponível em: https://www.microsoft.com/en-us/hololens/buy. Acesso em: 28 maio 2022.

MILGRAM, P.; KISHINO, F. A Taxonomy of Mixed Reality Visual Displays. *IEICE Trans. Information Systems*, vol. E77-D, no. 12, p. 1321–1329, dez. 1994.

MOSER, K. R.; SWAN, J. E. Evaluation of user-centric optical see-through head-mounted display calibration using a leap motion controller. In: *2016 IEEE Symposium on 3D User Interfaces (3DUI)*. [s.n.], 2016. p. 159–167. Disponível em: https://ieeexplore.ieee.org/document/7460047. Acesso em: 03 jan. 2023.

NISHIDA, J. *MediaPipe Unity Plugin*. 2022. Disponível em: https://github.com/homuler/MediaPipeUnityPlugin. Acesso em: 03 jan. 2023.

OUDAH, M.; AL-NAJI, A.; CHAHL, J. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques. *Journal of Imaging*, v. 6, n. 8, 2020. ISSN 2313-433X. Disponível em: https://www.mdpi.com/2313-433X/6/8/73. Acesso em: 03 jan. 2023.

PEDDIE, J. *Augmented Reality: Where We Will All Live*. 1. ed. Basel, Switzerland: Springer International Publishing, 2017. ISBN 978-3-319-54502-8. Disponível em: https://link.springer.com/book/10.1007/978-3-319-54502-8. Acesso em: 03 jan. 2023.

PETRICK, E. R. A Historiography of Human–Computer Interaction. *IEEE Annals of the History of Computing*, v. 42, n. 4, p. 8–23, 2020. Disponível em: https://ieeexplore.ieee.org/abstract/document/9140327. Acesso em: 03 jan. 2023.

PINHO, M. S.; CORRêA, C. G.; NAKAMURA, R.; JR, J. L. B. Técnicas de Interação. In: TORI, R.; HOUNSELL, M. d. S. (Ed.). *Introdução a Realidade Virtual e Aumentada*. 3. ed. Porto Alegre: Editora SBC, 2020. p. 165–195. ISBN 978-65-87003-54-2. Disponível em: https://doi.org/10.5753/sbc.6654.2. Acesso em: 03 jan. 2023.

PIUMSOMBOON, T.; CLARK, A.; BILLINGHURST, M.; COCKBURN, A. User-Defined Gestures for Augmented Reality. In: *CHI '13 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2013. (CHI EA '13), p. 955–960. ISBN 978-1-4503-1952-2. Disponível em: https://doi.org/10.1145/2468356.2468527. Acesso em: 03 jan. 2023.

POCO. *POCO X3 NFC Specs*. 2020. Disponível em: https://www.po.co/global/poco-x3-nfc/specs/. Acesso em: 03 jan. 2023.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de Software: Uma Abordagem Profissional*. 9. ed. [S.l.]: AMGH, 2021. ISBN 978-6558040101.

PTC. *Vuforia Documentation*. 2022. Disponível em: https://library.vuforia.com/. Acesso em: 03 jan. 2023.

REIFINGER, S.; WALLHOFF, F.; ABLASSMEIER, M.; POITSCHKE, T.; RIGOLL, G. Static and Dynamic Hand-Gesture Recognition for Augmented Reality Applications. In: JACKO, J. A. (Ed.). *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 728–737. ISBN 978-3-540-73110-8. Disponível em: https://link.springer.com/chapter/10.1007/978-3-540-73110-8_79. Acesso em: 03 jan. 2023.

ROMPAPAS, D.; RODDA, C.; BROWN, B. C.; ZERKIN, N. B.; CASSINELLI, A. Project Esky: an Open Source Software Framework for High Fidelity Extended Reality. In: *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2021. (CHI EA '21), p. 1–6. ISBN 978-1-4503-8095-9. Disponível em: https://doi.org/10.1145/3411763.3451804. Acesso em: 25 maio 2022.

SANTOS, J. dos. *Review Poco X3 NFC: Uma opção da Xiaomi que ainda vale a pena*. 2021. Disponível em: https://canaltech.com.br/produtos/review-poco-x3-nfc-186808/. Acesso em: 03 jan. 2023.

SINHA, G.; SHAHI, R.; SHANKAR, M. Human Computer Interaction. In: *2010 3rd International Conference on Emerging Trends in Engineering and Technology*. [s.n.], 2010. p. 1–4. Disponível em: https://ieeexplore.ieee.org/abstract/document/5698279. Acesso em: 03 jan. 2023.

SKARBEZ, R.; SMITH, M.; WHITTON, M. C. Revisiting Milgram and Kishino's Reality-Virtuality Continuum. *Frontiers in Virtual Reality*, v. 2, 2021. ISSN 2673-4192. Disponível em: https://www.frontiersin.org/articles/10.3389/frvir.2021.647997. Acesso em: 03 jan. 2023.

TECHNOLOGIES, U. *Unity - Manual: Unity User Manual 2021.3 (LTS)*. 2022. Disponível em: https://docs.unity3d.com/2021.3/Documentation/Manual/UnityManual.html. Acesso em: 03 jan. 2023.

TURNER, A.; COULTER, D. *Coordinate systems in DirectX - Mixed Reality*. 2021. Disponível em: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/native/coordinate-systems-in-directx. Acesso em: 03 jan. 2023.

ULTRALEAP. *Leap Motion Controller Data Sheet*. 2020. Disponível em: https://www.ultraleap.com/datasheets/Leap_Motion_Controller_Datasheet.pdf. Acesso em: 25 maio 2022.

YONG, M. G. *Empowering Live Perception with MediaPipe*. California: [s.n.], 2019. Disponível em: https://files.devnetwork.cloud/AIDevWorld/presentations/2019/Ming_Guang_Yong.pdf. Acesso em: 03 jan. 2023.

ZHANG, F.; BAZAREVSKY, V.; VAKUNOV, A.; TKACHENKA, A.; SUNG, G.; CHANG, C.-L.; GRUNDMANN, M. MediaPipe Hands: On-device Real-time Hand Tracking. jun. 2020. Disponível em: http://arxiv.org/abs/2006.10214. Acesso em: 03 jan. 2023.