Interfaces Naturais para Realidade Virtual: Prototipação com Tecnologias de Visão Computacional

Lucas Pereira Salbego[1], **Aline de Campos**[1]

[1] Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)

lucas.salbego99@edu.pucrs.br, aline.campos@pucrs.br

Resumo. *Este trabalho propõe o desenvolvimento de uma solução para a interação em ambientes virtuais utilizando apenas uma webcam. A realidade virtual tem ganhado espaço em várias áreas no Brasil, mas o alto custo dos dispositivos HMD limita seu uso pessoal. Este projeto visa criar uma solução acessível, permitindo que usuários interajam com ambientes virtuais usando suas mãos e cabeça, sem a necessidade de capacetes especializados. A proposta envolve o desenvolvimento de uma aplicação que colete dados de movimento através de uma câmera e os transmita para aplicações VR. A solução utiliza tecnologias como OpenCV para captura de imagem, MediaPipe para processamento de movimento e OpenVR para comunicação com aplicações VR. O objetivo é proporcionar uma alternativa open-source para interação com ambientes virtuais, ampliando o acesso à realidade virtual e demonstrando o potencial de soluções similares.*

Abstract. *This project proposes the development of a solution for interaction in virtual environments using only a webcam. Virtual reality has been gaining traction in various fields in Brazil, but the high cost of HMD devices limits their personal use. This project aims to create an affordable solution, allowing users to interact with virtual environments using their hands and head, without the need for specialized headsets. The proposal involves developing an application that collects motion data through a camera and sends it to VR applications. The solution utilizes technologies such as OpenCV for image capture, MediaPipe for motion processing, and OpenVR for communication with VR applications. The goal is to provide an open-source alternative to traditional interaction in virtual environments, expanding access to virtual reality and demonstrating potential in similar solutions.*

1. Introdução

A realidade virtual e todas as tecnologias relacionadas vem interessando o mundo cada vez mais desde 2010, com o desenvolvimento e eventual lançamento do Oculus Rift, um capacete de realidade virtual, ou HMD *(Head Mounted Display, em inglês)*, que trouxe inovações como menor distorção de imagem e um maior campo de visão comparado às soluções da época, além disso a intenção de disponibilizar um HMD moderno ao publico gerou muita expectativa e investimento, resultando na aquisição da empresa pelo Facebook (atual Meta) em 2014 (G1, 2014).

No Brasil, essas tecnologias vêm ganhando cada vez mais tração nas áreas profissionais, de educação, e pesquisa e desenvolvimento. Neste ano, por exemplo, houve um grande investimento por parte do Estado de Goiás em construir o maior laboratório de tecnologias imersivas do país (Carnevalli, 2024), e na área de educação profissional, já é possível ver o uso de realidade virtual para a capacitação de funcionários, como é o caso da Receita Federal, que aplica essa prática desde o ano passado para combater o contrabando (Brasil, 2023).

Com isso em mente, o uso pessoal desses dispositivos ainda não é acessível o suficiente para grande parte da população do país por se tratar de uma tecnologia nova e de alto custo, ao mesmo tempo que há um interesse cada vez maior pela interação virtual e um entusiasmo com essas tecnologias no país (Calliari, 2022).

Após o período de isolamento por conta do COVID-19, práticas como o trabalho e educação remota cresceram e se tornaram válidas e mais eficientes, e a nível global, grandes nomes da tecnologia como a Meta estão trabalhando em unir a realidade virtual a instituições de ensino (Duffy, 2024).

Isso gera situações em que pessoas gostariam de interagir em um ambiente virtual mas não tem os recursos necessários, ou o hardware existente pode não se adaptar a eles. Para isso, seria necessário um meio de interação que não dependesse de um capacete especializado, e que permitisse ao usuário interagir nesses ambientes usando dispositivos amplamente disponíveis e de sua preferência.

1.1. Definição do problema

Surgem algumas complicações ao interagir com dispositivos HMD. Primeiramente, o alto custo. Há uma resistência na adoção desse tipo de tecnologia por conta disso, e esse efeito se intensifica em países subdesenvolvidos.

Dispositivos imersivos podem não se adequar a certas pessoas e situações. O problema mais comum é a náusea, gerada pelo contraste entre o ambiente virtual e o contexto físico do usuário (Raaflaub & Andina, 2023), que pode ser mais agravante em determinados indivíduos. Além disso, o usuário pode não conseguir, ou apenas não se sentir confortável vestindo um dispositivo relativamente pesado em sua cabeça. Além dessas

dificuldades, o próprio desenvolvimento de aplicações em VR (*Virtual Reality*, em inglês) sem um dispositivo em mãos não é ideal, principalmente em um contexto de teste de usabilidade.

Para tanto, espera-se criar uma alternativa *open-source* para interação humano-computador em ambientes de realidade virtual, trazendo uma proposta aplicada a interfaces naturais. Ela deve ser capaz de permitir ao usuário interagir com um ambiente virtual usando as mãos e a cabeça, necessitando apenas de um computador e uma câmera.

1.2. Objetivos

Apresenta-se como principal objetivo construir uma aplicação capaz de permitir a interação em ambientes virtuais usando uma câmera como entrada. Ela deve coletar os dados posicionais de pontos específicos do usuário, mapear os movimentos e gestos detectados em ações, e ser capaz de ser reconhecida corretamente em uma aplicação VR, assim permitindo sua operação.

A aplicação deverá utilizar as mãos do usuário como forma principal de interação, sendo assim uma implementação conceituada em interfaces naturais. Assim, foram estabelecidos três objetivos específicos para nortear o desenvolvimento da aplicação:

a) coletar dados de captura de movimento através de uma câmera, assumindo que ela se encontra acima de uma tela ou integrada em um *laptop*;

b) implementar um *driver* de dispositivo, de tal modo a sugerir que há um HMD compatível conectado;

c) integrar os dados ao driver em tempo real, permitindo o mapeamento de movimentos e gestos do usuário em ações como pressionar botões ou gatilhos em aplicações existentes.

2. Fundamentação teórica

A seguir serão apresentados fundamentos relevantes para a compreensão da concepção e desenvolvimento deste projeto. Inicia-se pela definição de interfaces naturais de usuário, o conceito de realidade estendida e seus subtipos, bem como dispositivos relacionados a realidade virtual.

2.1. Natural User Interface (NUI)

As Interfaces Naturais de Usuário (em inglês, *Natural User Interface - NUIs*) são com foco em interação com os usuários de maneira intuitiva com uso de gestos com as mãos, fala, tato ou qualquer outro comportamento natural humano. Elas buscam minimizar a curva de aprendizado tornando a interação com a tecnologia mais semelhante às interações cotidianas (Jacob *et al*., 2008).

A principal característica destes sistemas se concentra na não necessidade de dispositivos intermediários comumente usados em interações digitais, tais como teclados e

mouses. Existem muitas formas de implementação destes recursos, sendo possível fazer desde uma câmera até sensores avançados. Estão nesta categoria os *smartphones* e outros dispositivos móveis com tecnologia *touchscreen*, uma vez que permitem uma interação direta com os elementos da tela (Tanimura & Ueno, 2013). Além disso, há crescente adoção de sistemas baseado em comandos por voz, tais como Alexa e Google Assistant.

2.2. Virtual Reality (VR)

Virtual Reality, em português, Realidade Virtual, trata-se de do uso de tecnologias para criar ambientes imersivos simulando mundos reais ou imaginários. O conceito nasce nos 60, juntamente com o desenvolvimentos de dispositivos físicos alternativos para interação com simulações, tais como luvas, controladores manuais e outros recursos com integração de tato e voz. Outros termos como "realidade artificial"e "ciberespaço"foram usados para descrever essas estruturas que inicialmente tinham foco em experiências sensoriais e entretenimento (Wohlgenannt *et al*., 2020).

Ao longo do tempo e das novas possibilidades de recursos tecnológicos, outros conceitos correlatos surgiram, como a *Augmented Reality* (AR), ou Realidade Aumentada e *Mixed Reality* (MR) ou Realidade Mista. Atualmente pode-se considerar os três conceitos (VR, AR e MR) como parte do campo da *Extended reality* (XR), ou Realidade Estendida (Figura 1) (Mann *et al*., 2018).

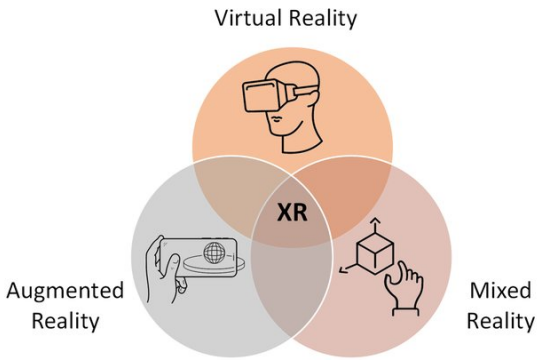

Figura 1. Realidade Estendida (Extended Reality - XR)

Os recursos que implementam Realidade Aumentada são compreendidos como camadas digitais em cenários e processos do mundo físico, de forma a potencializar a percepção do usuário. É bastante adotado através de *smartphones* combinando visões de elementos físicos e virtuais para auxilio em localizações, comunicação, entretenimento e *marketing* (Mann *et al*., 2018).

Já a Realidade Mista, está relacionada a possibilidade de além da existência de elementos físicos e virtuais em uma mesma estrutura, esses poderem interagir entre si. Estes recursos tem sido bastante explorados em processos que envolvem simulações (Tamura *et al*., 2001) .

2.3. Head-mounted Display (HMD)

O dispositivos *Head-mounted Display* têm suas origens no conceito de *display* interativo. Rolland e Hua (2005) os descreve como "dispositivos de exibição que seguem a linha de visão do usuário, permitindo não apenas movimento livre da cabeça, mas também, potencialmente e de forma importante, mobilidade total do corpo [..] uma solução simples seria vestir o dispositivo" (p. 1).

Sendo assim, compreende-se os HMD como telas acopladas ao rosto de alguma forma, permitindo movimentos e processos de imersão. Atualmente pode-se associar este conceito aos capacetes de realidade virtual (*Virtual Reality - VR*) que tem sido incorporados a diversos processos, como o Oculus Rift, Meta Quest, entre outros.

Embora nos últimos anos tenha havido uma grande visibilidade destes recursos por conta das ascensão de plataformas do chamado Metaverso, esse tema vem sendo trabalhado desde os anos 60, com o pioneiro trabalho do pesquisador Ivan Sutherland (Sutherland, 1968). O cientista desenvolveu um sistema complexo de sensores presos ao teto e *displays* de raios catódicos acoplados no que seria o primeiro dispositivo HMD, permitindo que o usuário veja um ambiente em três dimensões que considera sua posição espacial (Figura 2).

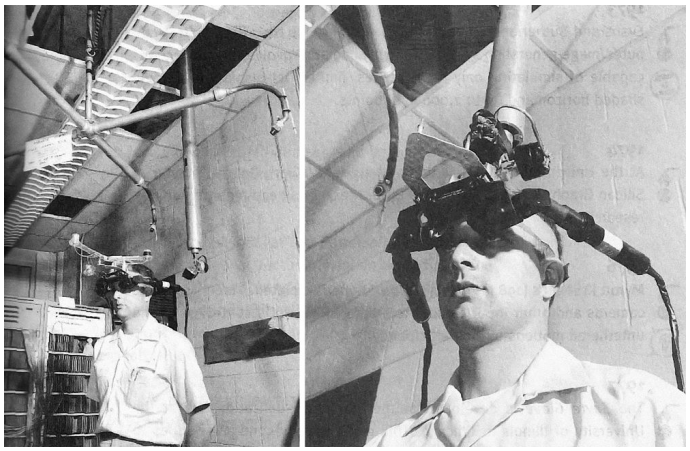

Figura 2. Cientista Ivan Sutherland em 1968 com o protótipo de HMD.

Com o passar dos anos e os avanços em termos de hardware e capacidade de processamento, o potencial de aplicação dos HMD tem se mostrado versátil, sendo bastante adotado em aplicações para entretenimento e comunicação. Também tem sido aplicado com foco em simulações, como treinamento em âmbito militar, práticas médicas, processos industriais e até mesmo em cenários educacionais com intuito de aumentar o engajamento e imersão dos estudantes (Jin *et al.*, 2022). Entretanto, ainda são enfrentadas barreiras relacionadas ao custo e possibilidade acessibilidade destes dispositivos, o que se mostra como obstáculos que ainda demandam soluções para sua total popularização em diferentes cenários.

3. Trabalhos relacionados

Existe uma grande quantidade de produção relevante na área de Realidade Virtual. As pesquisas atuais apresentam o uso de câmeras e captura de movimento a um ambiente VR apropriado, com ou sem um dispositivo HMD em mãos e são experiências interessantes ao desenvolvimento deste trabalho.

Murugana *et al*. (2024) apresenta um sistema de rastreamento de mãos para interações em RV e RA que faz uso de uma *webcam*. O intuito dos pesquisadores foi evitar a limitação do posicionamento das mãos que pode causar desconforto ao usuário, oferecendo a possibilidade de posturas mais naturais. Para demonstrar o funcionamento desenvolveram uma aplicação em Unity3D com a capacidade do sistema de replicar articulações precisas das mãos e realizar tarefas como pressionar botões e empilhar cubos.

Já Yang *et al*. (2022) propõem a aplicação HybridTrak, que possibilita o rastreamento corporal completo em tempo real através de captura por webcam. A plataforma adota redes neurais para converter e transformar poses corporais em 2D para 3D o que em situações de realidade virtual pode melhorar a sensação de presença e a possibilidade de interações por meio de gestos corporais.

O trabalho de Esmaeilabadi (2012) apresenta a proposição de uma interface interativa para experiências em realidade virtual onde há um sensor de visão computacional com uma webcam monocular de resolução baixa onde os movimentos da cabeça são capturados, analisados e traduzidos para adaptar ao ângulo de visão relativo em tempo real. E se assemelha mais a aplicação proposta por não usar um *display* montado e capturar o movimento da cabeça do usuário.

Além de referências acadêmicas, buscou-se também por aplicações comerciais existentes ao encontro da proposta deste projeto. Destaca-se o *Mock HMD XR Plugin*[1] da Unity, que simula um dispositivo HMD em nível de *software*, porém não oferece suporte ao uso de *webcam* ou qualquer *hardware* por si só, sendo necessário que os desenvolvedores controlem o HMD simulado através de código.

O *VRidge*[2] é um aplicativo capaz de transformar o *smartphone* do usuário em um HMD genérico, usando o giroscópio para movimentar a cabeça do usuário em VR. Já o *Windows Mixed Reality Simulator*[3] apresenta uma funcionalidade incorporada ao sistema operacional Windows que permite ao usuário usar o teclado e mouse para simular um dispositivo HMD dentro de algumas aplicações. A maioria das soluções usam o teclado e mouse e não são capazes de trabalhar com *hand-tracking*, ou seja, a detecção de movimentos das mãos do usuário. Logo, entende-se que há espaço para explorar aplicações que acima de tudo, permitam o controle das mãos usando uma *webcam*. Com isso em

[1]Disponível em: https://docs.unity3d.com/Packages/com.unity.xr.mock-hmd@1.0/manual/index.html
[2]Disponível em: https://www.riftcat.com/
[3]Disponível em: https://learn.microsoft.com/pt-br/windows/mixed-reality

mente, o valor da proposta está em compilar conhecimentos existentes e transmitir todos dados de captura de uma vez só da melhor forma possível, de tal modo que aplicações em VR reconheçam um HMD conectado e isso permita o amplo uso das mesmas a usuários que não possuem dispositivos compatíveis com VR.

4. Aspectos metodológicos

Com o objetivo de aprofundar os conhecimento na implementação de aplicações para realidade virtual e de desenvolver um projeto prático, optou-se pela construção de uma prova de conceito que demonstre o potencial da alternativa proposta gerada através de um experimento prático (Kendig, 2015).

Para alcançar esses objetivos, foi adotada a prática de prototipação Houde e Hill (1997). Assim, foram desenvolvidos protótipos incrementais como forma de dividir o trabalho em etapas que ao final resultem em um experimento de maior fidelidade. Inicialmente buscou-se desenvolver um protótipo conceitual, como um fluxograma que representa a implementação real, mas sem aspectos detalhados e funcionalidades. Em seguida foram implementados protótipos gradualmente mais fiéis à solução, para ao fim chegar-se a um protótipo funcional o suficiente para servir como prova de conceito.

5. Desenvolvimento do projeto

A seguir apresenta-se o processo de desenvolvimento com as pesquisas preliminares e experimentações de recursos, indicando os resultados obtidos, mudanças necessárias e definições realizadas em cada etapa a partir de um processo de prototipação centrado na compreensão teórica e técnica e na curva de aprendizado com vistas a solução proposta.

5.1. Pesquisa preliminar

Primeiramente, foi feita uma pesquisa sobre as diferentes maneiras de implementar dispositivos interativos virtuais em uma aplicação, como um ponto de partida para compreender a direção do projeto. Durante a pesquisa foram consideradas três alternativas:

a) *Windows HID Driver*: criação de um driver Windows para dispositivos interativos, uma alternativa válida, porém não se trata de uma ferramenta própria para o contexto de realidade virtual e sua configuração inicial seria a mais complexa;

b) *OpenHMD Driver*: uma biblioteca para integração VR inteiramente *open-source*, alternativa que, apesar de ser válida, envolveria o desenvolvimento de uma aplicação para demonstração e testes, pois se trata de uma biblioteca sem suporte nativo a jogos e aplicativos existentes;

c) *OpenVR Driver*: uso da API da Valve para criação de drivers para realidade virtual, que apresentou-se promissora por ser uma solução adotada em todas as aplicações existentes na plataforma Steam, a maior em jogos e aplicações de realidade virtual para computadores. Além disso, a documentação e códigos de exemplo disponíveis fornecem uma base sólida para construção dos protótipos.

Sendo assim, optou-se por desenvolver um protótipo em cima da interface OpenVR.

5.2. Primeiro protótipo: fluxo básico de operação

Primeiramente foi analisada a estrutura geral do OpenVR e da sua integração com as aplicações. A API é segmentada em Aplicação e Driver - A API de Aplicação define como jogos e aplicativos devem se comunicar com os drivers, coletando dados referentes aos dispositivos de entrada como controles e HMDs.

O segundo segmento, que foi de fato utilizado, descreve como as fabricantes desses dispositivos devem expôr seus dados para a aplicação e gerar um driver no formato DLL *(Dynamic Link Library)*. Um arquivo nesse formato é carregado dinamicamente em memória e é utilizado para modularizar segmentos de código, assim a aplicação é capaz de carregar apenas as funcionalidades que precisa. No contexto de um driver de dispositivo, também é esperado que o mesmo seja sucinto para reduzir a latência durante o uso.

Com isso, decidiu-se por separar o protótipo em duas grandes etapas. Primeiro, em uma aplicação separada, será coletado um conjunto de dados de rastreamento corporal do usuário, esses dados serão processados e enviados através de uma IPC *(Inter-Process Communication)* para os drivers.

Com esses dados disponíveis, será uma questão de implementar os drivers necessários para replicar os movimentos do usuário em um ambiente VR. Os movimentos da cabeça, por exemplo, são descritos em um driver de HMD nas mesmas condições de posição e rotação capturadas. Separando o protótipo dessa forma, buscou-se reduzir a carga dos drivers executando as etapas mais complexas em um processo independente (Figura 3).

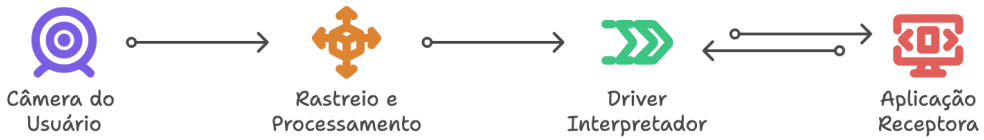

Figura 3. Fluxo inicial do projeto.

O fluxo de operação se inicia com o acesso à câmera do usuário, nessa etapa optou-se pelo OpenCV por se tratar de uma biblioteca de visão computacional amplamente difundida, assim foi possível ter acesso a um material de qualidade para consulta. A próxima etapa envolve a coleta de marcadores - pontos que correspondem a regiões específicas do corpo, como a ponta do nariz ou os olhos do usuário.

Definiu-se trabalhar com o Google MediaPipe para esse propósito, mais especificamente, com suas soluções de modelos de aprendizado de máquina pré-treinados em detecção de marcadores. Também havia a possibilidade de usar o OpenCV para detecção de marcadores, mas decidiu-se pelo MediaPipe por se tratar de uma solução modular, robusta e de fácil uso. Assim há a possibilidade de executar múltiplos modelos especializados em detecção de partes específicas do corpo simultaneamente.

A terceira etapa, após o acesso a câmera e a coleta de marcadores, é o processamento e envio de dados. Existem diversas abordagens de IPC disponíveis, mas por questões de familiaridade e por se tratar de um protótipo inicial foram utilizados *sockets* de rede UDP para a comunicação.

5.3. Segundo protótipo: captura de movimentos simples da cabeça

O começo da fase de implementação envolve a configuração inicial do OpenCV para o acesso à câmera, e da inicialização e testes práticos com o MediaPipe. O MediaPipe, como o nome sugere, é um framework de pipelines para processamento de mídia, a sua execução envolve o envio de pacotes de dados ao longo de um grafo de processamento onde nodos produzem e consomem dados.

A proposta é promover modularização e paralelismo separando o processamento de imagem e vídeo em nodos que são retirados da memória assim que concluem sua contribuição no processo. As soluções escolhidas foram o *MediaPipe Hands* e *MediaPipe Face Mesh*, ambas utilizam a estrutura de grafos para executar modelos baseados em redes neurais convolucionais (CNNs) e permitir a captura de marcadores estimados nas mãos e no rosto do usuário.

5.3.1. Captura e envio de marcadores

A saída dos modelos é uma lista de marcadores e suas posições, representadas em um sistema de coordenadas tridimensional. A maioria das soluções do MediaPipe possuem duas opções de sistemas de coordenadas - uma dada em metros de distância da câmera chamada de *global landmarks* ou marcadores globais, e outra normalizada, onde o ponto (0,0,0) representa uma coordenada no canto inferior esquerdo da tela, enquanto (1,1,0) descreve o canto superior direito.

O componente z das coordenadas normalizadas representa a profundidade do marcador, quanto menor o valor, mais próximo se estima que o usuário esteja da câmera, contudo, não foi possível compreender sua obtenção nem escala exatas, por conta disso definiu-se por não trabalhar com a profundidade normalizada se possível. Infelizmente, o modelo de detecção facial não possui marcadores globais, logo, foi limitado o escopo de uso do protótipo assumindo que o usuário não se movimenta ao longo do eixo z.

Após a coleta dos marcadores estimados, foi preciso desenvolver um sistema de envio de dados através de *sockets*. O processo envolve a formação do pacote de dados UDP a ser enviado. O objetivo desse teste inicial é permitir movimentos simples para validação da estrutura, logo, o pacote enviado consiste apenas nas coordenadas da ponta do nariz por enquanto. Foi desenvolvido o pacote como uma *string* no formato [*<rótulo>: (x,y,z), <rótulo>: (x,y,z) ...*], usando os rótulos para navegar pelo *buffer* de dados, realizar o *parsing*, e coletar as coordenadas correspondentes.

5.3.2. Criação do driver

A criação de um *driver* de dispositivo demanda tempo. Felizmente, a equipe do OpenVR disponibiliza diversos exemplos completos de drivers para consulta, por conta disso, foi possível priorizar esforços no entendimento da estrutura geral e na adição de funcionalidades específicas ao protótipo (Figura 4). .

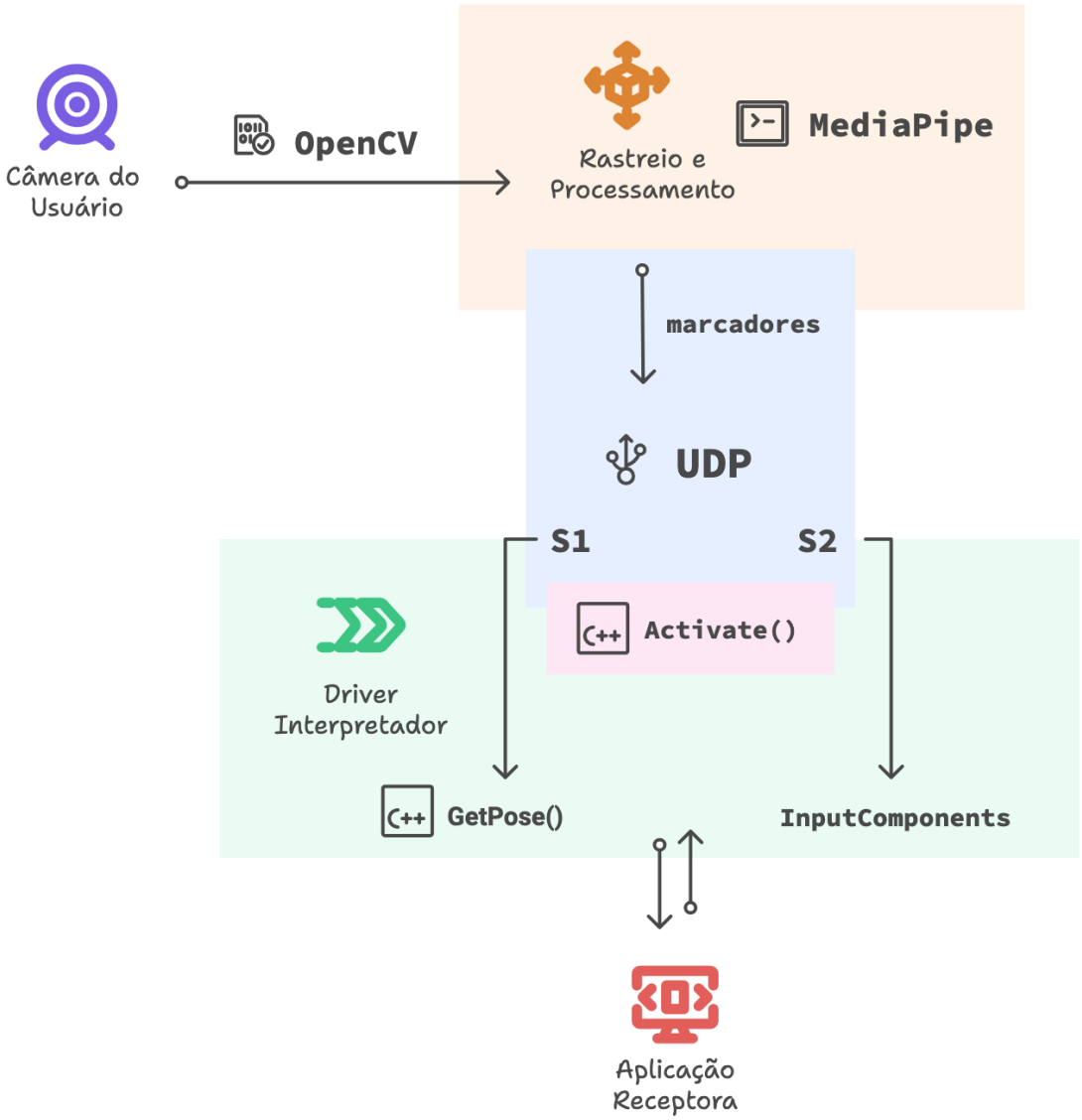

Figura 4. Esquema conceitual da solução proposta.

A estrutura de um *driver* nessa API consiste em, pelo menos, a criação de três classes: Uma fábrica de drivers *DriverFactory*, uma provedora de dispositivos *IServerTrackedDeviceProvider*, e uma classe que contém a implementação de cada dispositivo *ITrackedDeviceServerDriver* (Valve, 2022). A fábrica define a criação de toda a estrutura do driver e é chamada pelo *runtime* após a DLL ser detectada no diretório. A próxima classe na hierarquia é a provedora de dispositivos. Ela cumpre um papel semelhante à fabrica, mas

no escopo interno do driver, e é responsável por inicializar todos os dispositivos como controles e HMDs que deseja-se comportar. Por fim apresentam-se as classes principais de cada dispositivo. Elas possuem uma interface consideravelmente maior do que as anteriores é onde o maior foco de atenção será posicionado.

No momento, as funções essenciais eram a ***Activate(unObjectId)*** e ***GetPose()***. A função *Activate()* é chamada pela classe provedora apenas uma vez assim que o dispositivo é criado e serve para inicializar dados como nome, número de série e outras informações estáticas do dispositivo, além de chamar funções que precisam ser executadas apenas uma vez. Seu parâmetro *unObjectId* representa o identificador único do dispositivo e deve ser gerado pela *IServerTrackedDeviceProvider*. A função *GetPose()* é onde dados como a posição e rotação atuais são atualizados. Ela é chamada idealmente quando novos dados estão disponíveis, contudo, a atualização desses dados fica a critério do desenvolvedor.

O primeiro passo após a configuração inicial do driver é a conexão ao socket UDP e a criação do parser de dados. A inicialização do socket (S1) acontece no começo da função *Activate*, e o parser é chamado dentro da função *GetPose()*, onde o vetor de posição é atualizado com os dados coletados.

O parser foi implementado usando a função *strtok* da biblioteca padrão do C++, que divide um *buffer* em *tokens* através de delimitadores. Essa função coleta um *token* de cada vez a partir de um ponteiro passado como parâmetro, e ao ser chamada em sequência com um ponteiro nulo, continua do ponto onde o último *token* foi coletado. A seguir é possível visualizar um trecho de cada uma das funções, com detalhes omitidos a favor da clareza.

Código-fonte 1. Trecho de implementação

```
parseLandmark(buffer, id) {
 ...
 next_token = strstr(buffer, id);

 result.x = strtof(strtok_s(next_token, ": [ ", &next_token), NULL);
 result.y = strtof(strtok_s(NULL, " ", &next_token), NULL);
 result.z = strtof(strtok_s(NULL, " ]", &next_token), NULL);

 return result;
}

Activate(unObjectId) {

 udpSocket = setupSocket();
 if (udpSocket == INVALID_SOCKET) {
    DriverLog("Socket setup failed.\n");
 }

  device_index_ = unObjectId;
  ...
}
```

A função *receivePoseCoordinates* presente no trecho abaixo foi criada para chamar o parser, coletar os marcadores e converter os dados para o formato esperado, retornando um vetor com todos os marcadores disponíveis. O primeiro elemento desse vetor é a ponta do nariz, que será usada para definir a posição atual da cabeça. Ressalta-se que não foram trabalhadas rotações neste momento, logo, a ponta do nariz serve como uma boa estimativa de posição da cabeça considerando que o usuário geralmente mantém o rosto direcionado para o monitor.

Código-fonte 2. Trecho de implementação

```
GetPose() {
  ...
  poseCoordinates = receivePoseCoordinates(udpSocket);

  if (!poseCoordinates.empty()) {
      currentPosition = poseCoordinates[0];
  }

  pose.vecPosition[0] = currentPosition.x;
  pose.vecPosition[1] = currentPosition.y;
  pose.vecPosition[2] = currentPosition.z;
  ...
}
```

A esse ponto criou-se uma aplicação que estima, coleta e envia marcadores para um *socket*, bem como um *driver* que lê esses dados e atualiza a posição do usuário em tempo real. Com isso, é possível movimentar a cabeça lateralmente e verticalmente em aplicações VR (Figura 5).

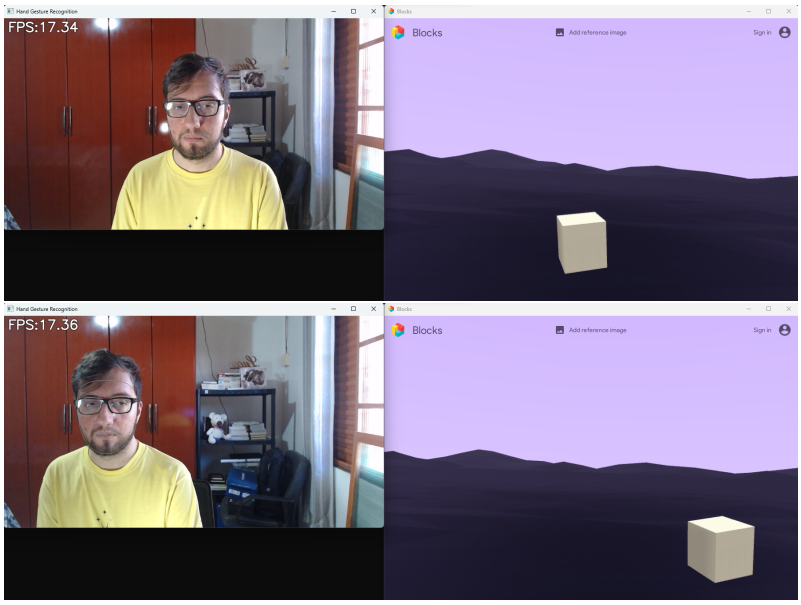

Figura 5. Movimento lateral em VR

5.4. Terceiro protótipo: Implementando rotações em 3D

Para implementar rotações, é necessário entender como elas são representadas. Com o trabalho inovador de Hamilton (1844), foi-se introduzido um novo sistema numérico chamado de Quatérnios, uma extensão dos números complexos representada na forma:

$q = a + bi + cj + dk$

onde:

a, b, c são números reais (também chamados de escalares)

i, j, k números complexos.

Para representar rotações se utiliza um quatérnio específico chamado de *quatérnio unitário*, que tem sua magnitude $|q| = 1$. A parte imaginária de um quatérnio unitário pode ser abstraída como os componentes x, y, z de um vetor unitário tridimensional, esse vetor representa o eixo de rotação descrito pelo quatérnio, e os valores escalares representam a magnitude da rotação em si. Com esse sistema é possível representar rotações em 3D de maneira eficiente, evitando o problema de *gimbal lock*, a perda de um grau de liberdade quando dois eixos de rotação se alinham. O conceito de quatérnios foi incorporado na computação gráfica pela primeira vez nos anos 80 conforme o trabalho de Shoemake (1985) e mais adiante em áreas como a indústria aeroespacial, como apresenta o trabalho de Markley (1999). Atualmente, é também o sistema presente no OpenVR.

As principais vantagens do uso de quatérnios em computação gráfica são a maior eficiência e prevenção de *gimbal lock* durante processo de interpolação, logo, não é preciso trabalhar com quatérnios durante todo o processo. A obtenção da rotação atual da cabeça utiliza outras representações de rotações mais tradicionais comomatrizes de rotação, que serão submetidas a uma conversão para quatérnios ao final.

Para o cálculo de rotação fez-se uso de quatro marcadores: O queixo, o centro da testa, e os dois olhos. Assim foi possível construir dois vetores que representam as direções para cima (centro da testa - queixo) e para direita (olho direito - olho esquerdo) em relação a cabeça. Com esses vetores foi possível obter um terceiro vetor usando o produto vetorial, esse representa a direção em que a cabeça do usuário está voltada, e ao normalizar esses vetores, uma *base ortonormal* foi obtida. Essa é a chave para o cálculo de rotação, pois uma base ortonormal é equivalente a uma matriz de rotação quando os vetores representam as colunas, e após a construção da matriz, pode-se aplicar uma das fórmulas de conversão para quatérnios de rotação, finalizando o processo.

Com a rotação de cada *frame*, basta atualizar o quatérnio de rotação na função *GetPose()*, e o *runtime* ficará encarregado de calcular a interpolação entre a posição atual e a do *frame* anterior. Agora a rotação do usuário no ambiente virtual corresponde a rotação real da cabeça (Figura 6).

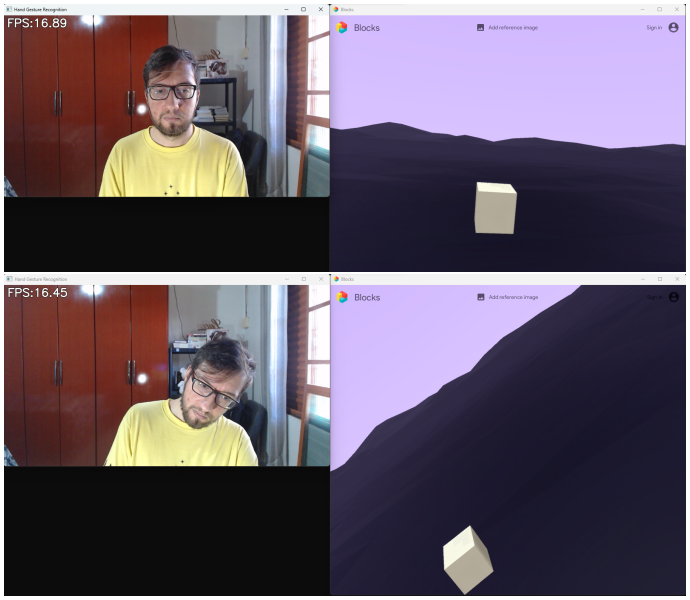

Figura 6. Rotação da cabeça em VR

5.5. Quarto protótipo: integração com movimentos gestuais

Recentemente, a Google disponibilizou uma solução de detecção de gestos na biblioteca do Mediapipe, porém, assim como nas outras soluções, ela utiliza um modelo pré treinado com uma lista pequena de gestos. Se for necessário modificar e treinar o modelo com novos dados, será preciso compreender a construção do pipeline em questão, contudo, apoiou-se na solução de Takahashi (2020) que implementou um modelo de reconhecimento de gestos usando o Mediapipe Hands e o disponibilizou junto dos recursos necessários para treina-lo com novos dados.

A solução utiliza o *Tensorflow* no desenvolvimento de uma rede *MLP (Multi-Layer Perceptron)* para a estimativa de gestos das mão e movimentos do dedo indicador. A rede se baseia nas posições dos marcadores estimados pelo *Mediapipe*, e durante o treinamento, associa um conjunto delas a um gesto específico. Com isso em mente, é possível gerar novos dados de treino registrando exemplos de marcadores e seu gesto associado no banco de dados da solução.

Os gestos escolhidos foram: apontar com o indicador, *pinch*, mão aberta e mão fechada. A proposta é mapear cada um dos gestos para ações correspondentes em um controle virtual, por exemplo, o *pinch* e a mão fechada podem ser mapeados como botões ou gatilhos, e o indicador (quando se faz o gesto de apontar) pode ser usado com movimentos de *swipe* para mapear as direções de um *joystick*.

O processo utilizado para obter os movimentos da cabeça foi expandido para coletar e enviar marcadores, rotações e o gesto atual das mãos através de outro socket (S2), que será consumido pelo driver do controle virtual para atualizar seus componentes de entrada (Figura 4). No momento optou-se por suportar a captura de uma mão por vez, pois o uso

do sistema atual com duas mãos gerava atrasos e seria preciso implementar paralelismo como uma solução para o problema.

O mapeamento dos botões envolveu habilitar componentes do controle virtual quando o gesto correspondente fosse detectado, diferente dos *joysticks*, que exigiu uma abordagem diferente. Foi necessário estimar a direção e intensidade do movimento do dedo indicador, para isso, é armazenado um histórico das últimas posições desse dedo enquanto o gesto de "apontar" é detectado. Foi feita a soma das diferenças dessa lista, e o vetor resultante é normalizado, representando a posição atual do *joystick* em uma escala (-1:1), onde o ponto [0,0] é o centro, [-1,-1] representa a posição inferior esquerda, e [1,1] a superior direita. Assim, gestos de *swipe* mais rápidos aceleram o *joystick* mais cedo, e gestos mais lentos possibilitam mais controle.

Figura 7. Exemplo de uso em que o gesto de fechar a mão é mapeado como gatilho para posicionar blocos no Google Blocks

5.6. Resultado da prova de conceito

Com o protótipo final, o usuário é capaz de mover a cabeça para visualizar a região ao seu redor e de mapear gestos para ações específicas, permitindo interações simples na aplicação escolhida. É possível usar o protótipo em qualquer aplicação que prevê suporte

ao OpenVR, e a escolha de coletar e processar dados em um processo independente permite a implementação futura de outros drivers para outras plataformas, com alterações mínimas no projeto.

Possíveis aplicações incluem o uso em capacitações para operação de máquinas, montagem de equipamentos, e salas de aula interativas, além do desenvolvimento de formas de interação acessíveis a aplicações VR existentes, já que o usuário tem a liberdade de mapear gestos a ações que façam mais sentido no seu contexto. Os principais aprendizados técnicos desse projeto incluem um maior entendimento sobre a criação de drivers, rotação de objetos usando quatérnios, e o uso de tecnologias de visão computacional.

6. Considerações finais

Com o desenvolvimento do protótipo final, foi possível desenvolver uma alternativa baseada em visão computacional para interação com sistemas de realidade virtual sem o uso de equipamentos específicos para VR. O processo de desenvolvimento envolveu a criação de vários protótipos com uma curva de aprendizado contínua, através do levantamento de tecnologias, consulta e entendimento de APIs, e criação de soluções.

Apesar do usuário conseguir mover a cabeça e visualizar um ambiente 3D, não há a possibilidade de virar o corpo além dos limites da câmera, logo, não há um meio de olhar o que está atrás do usuário. Uma solução possível seria o uso de um gesto seguido do movimento das mãos para controlar a câmera, similar aos métodos presentes no trabalho de Ganapathi e Sorathia (2023), em que se usaram gestos para explorar meios de locomoção em VR. Há uma perda de performance significativa vinda da ausência de paralelismo, já que atualmente todos os marcadores são estimados e processados na mesma *thread*. E como comentado na seção 5.5, a implementação de paralelismo também possibilitaria o uso de duas mãos simultâneas no projeto.

Durante o processo de prototipação, a captura de rotações da cabeça foi obtida usando operações vetoriais em pares de marcadores específicos, contudo, uma solução mais robusta seria a integração com o OpenFace - Um pacote de ferramentas de visão computacional com enfoque em operações e estimativa de elementos faciais.

Outra direção interessante é a integração do protótipo em motores gráficos como a *Unreal Engine*, *Unity* e *Godot*, com destaque à *Unity*, pela sua popularidade no desenvolvimento de aplicações para realidade virtual. Com isso, mais equipes seriam capazes de testar suas aplicações de maneira interativa, dispensando o uso de interfaces convencionais como mouse e teclado em situações onde o uso de *HMDs* e controles não é uma opção viável.

A prova de conceito demonstrou potencial para o desenvolvimento de formas de interação mais acessíveis, e há interesse em desenvolver o conceito para futuramente comportar uma vasta gama de gestos e mapeamentos criativos, que sirvam como novas alternativas de interação em ambientes virtuais para uma variedade maior de pessoas.

Referências

Brasil. (2023). Receita Federal inicia treinamentos com realidade virtual para combater o contrabando e descaminho. *Ministério da Fazenda - Receita Federal*. https://www.gov.br/receitafederal/pt-br/assuntos/noticias/2023/novembro/receita-federal-inicia-treinamentos-com-realidade-virtual-para-combater-o-contrabando-e-descaminho

Calliari, M. (2022). Ensino a distância, entretenimento digital e melhorias de home office animam brasileiros com o avanço do Metaverso. *IPSOS*. https://www.ipsos.com/pt-br/ensino-distancia-entretenimento-digital-e-melhorias-de-home-office-animam-brasileiros-com-o-avanco

Carnevalli, J. (2024). Goiás é pioneiro em tecnologias imersivas e realidade virtual. *Secretaria de Ciência, Tecnologia e Inovação do Governo de Goiás*.

Duffy, C. (2024). Mark Zuckerberg quer levar realidade virtual à sala de aula. *CNN Brasil*.

Esmaeilabadi, M. E. A. (2012). 3D Virtual Reality Navigation by Human's Head Movements Using a Generic Webcam. *Procedia Technology*, *3*, 121–131. https://www.sciencedirect.com/science/article/pii/S2212017312002423

G1. (2014). Facebook compra empresa de óculos de realidade virtual por US$ 2 bilhões. *G1*. https://g1.globo.com/tecnologia/noticia/2014/03/facebook-compra-empresa-de-oculos-de-realidade-virtual-por-us-2-bilhoes.html

Ganapathi, P., & Sorathia, K. (2023). User elicited gesture-based locomotion techniques for immersive VEs in a seated position: a comparative evaluation. *Frontiers in Virtual Reality*, *4*.

Hamilton, W. (1844). On quaternions; or a new system of imaginaries in algebra. *Philosophical Magazine*.

Houde, S., & Hill, C. (1997). Chapter 16 - What do Prototypes Prototype? Em M. G. Helander, T. K. Landauer & P. V. Prabhu (Eds.), *Handbook of Human-Computer Interaction (Second Edition)* (Second Edition, pp. 367–381). North-Holland. https://doi.org/10.1016/B978-044481862-1.50082-0

Jacob, R. J., Girouard, A., Hirshfield, L. M., Horn, M. S., Shaer, O., Solovey, E. T., & Zigelbaum, J. (2008). Reality-based interaction: a framework for post-WIMP interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 201–210.

Jin, Y., Ma, M., & Zhu, Y. (2022). A comparison of natural user interface and graphical user interface for narrative in HMD-based augmented reality. *Multimedia Tools Appl.*, *81*(4), 5795–5826. https://doi.org/10.1007/s11042-021-11723-0

Kendig, C. (2015). What is Proof of Concept Research and how does it Generate Epistemic and Ethical Categories for Future Scientific Practice? *Science and Engineering Ethics*, *22*. https://doi.org/10.1007/s11948-015-9654-0

Mann, S., Furness, T., Yuan, Y., Iorio, J., & Wang, Z. (2018). All Reality: Virtual, Augmented, Mixed (X), Mediated (X,Y), and Multimediated Reality. https://arxiv.org/abs/1804.08386

Markley, F. L. Aatborg University, Dept. of Control Eng. Colloquium. Em: Em *Spacecraft Attitude Representations*. 1999. https://ntrs.nasa.gov/citations/19990110711

Murugana, A. S., Behzada, A., Kim, E., Chung, J., & Jung, D. (2024). Development of webcam-based hand tracking for virtual reality interaction. *Fifth International Conference on Image, Video Processing, and Artificial Intelligence (IVPAI 2023)*. https://doi.org/10.1117/12.3023798

Raaflaub, C., & Andina, M. (2023). *Enjoo cibernético na realidade virtual: Por que ocorre e como pode ser tratado?* Swiss Info.

Rolland, J. P., & Hua, H. (2005). Head-Mounted Display Systems. *Encyclopedia of Optical Engineering*.

Shoemake, K. (1985). Animating rotation with quaternion curves. *SIGGRAPH Comput. Graph.*, *19*(3), 245–254. https://doi.org/10.1145/325165.325242

Sutherland, I. E. (1968). A head-mounted three dimensional display. *Fall Joint Computer Conference, Part I on - AFIPS'68 (Fall, Part I)*.

Takahashi, S. (2020, dezembro). *hand-gesture-recognition-using-mediapipe*. https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe

Tamura, H., Yamamoto, H., & Katayama, A. (2001). Mixed Reality: Future Dreams Seen at the Border between Real and Virtual Worlds. *IEEE Computer Graphics and Applications*, *21*, 64–70.

Tanimura, M., & Ueno, T. (2013). Smartphone User Interface. *Fujitsu Scientific and Technical Journal*, *49*(2), 227–230.

Valve. (2022). *OpenVR Driver Documentation*. https://github.com/ValveSoftware/openvr/wiki/Driver-Documentation

Wohlgenannt, I., Simons, A., & Stieglitz, S. (2020). Virtual Reality. *Business & Information Systems Engineering: The International Journal of WIRTSCHAFTSINFORMATIK*, *62*(5), 455–461. https://doi.org/10.1007/s12599-020-00658-

Yang, J. (, Chen, T., Qin, F., Lam, M. S., & Landay, J. A. (2022). HybridTrak: Adding Full-Body Tracking to VR Using an Off-the-Shelf Webcam. *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3491102.3502045