

**MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO**

**MATEUS CÂNDIDO LIMA DE CASTRO
JOÃO PEDRO DE ARAÚJO XAVIER**

**INTERAÇÃO POR *HAND-TRACKING* EM AMBIENTE DE REALIDADE
VIRTUAL**

**RIO DE JANEIRO
2020**

MATEUS CÂNDIDO LIMA DE CASTRO
JOÃO PEDRO DE ARAÚJO XAVIER

INTERAÇÃO POR *HAND-TRACKING* EM AMBIENTE DE REALIDADE
VIRTUAL

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador(es): Paulo Fernando Ferreira Rosa, Ph.D.
Jauvane Cavalcante de Oliveira, Ph.D.

Rio de Janeiro
2020

©2020

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro – RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

Castro, Mateus Cândido Lima de; Xavier, João Pedro de Araújo.

Interação por *Hand-Tracking* em ambiente de Realidade Virtual / Mateus Cândido Lima de Castro e João Pedro de Araújo Xavier. – Rio de Janeiro, 2020.

51 f.

Orientador(es): Paulo Fernando Ferreira Rosa e Jauvane Cavalcante de Oliveira.

Projeto de Final de Curso (graduação) – Instituto Militar de Engenharia, Engenharia da Computação, 2020.

1. Redes Neurais. 2. Aprendizado Profundo. 3. Realidade Virtual. 4. Sistema de Tempo Real. 5. Análise de Imagens. 6. Detecção de Pose. i. Rosa, Paulo Fernando Ferreira (orient.) ii. de Oliveira, Jauvane Cavalcante (orient.) iii. Título

**MATEUS CÂNDIDO LIMA DE CASTRO
JOÃO PEDRO DE ARAÚJO XAVIER**

**Interação por *Hand-Tracking* em ambiente de Realidade
Virtual**

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Bacharel em Engenharia da Computação.

Orientador(es): Paulo Fernando Ferreira Rosa e Jauvane Cavalcante de Oliveira.

Aprovado em Rio de Janeiro, 27 de outubro de 2020, pela seguinte banca examinadora:

Prof. Julio Cesar Duarte - D.Sc. do IME - Presidente

Prof. Paulo Fernando Ferreira Rosa - Ph.D. do IME

Prof. Jauvane Cavalcante de Oliveira - Ph.D. do LNCC

Prof. Luiz Carlos Pacheco Rodrigues Velho - Ph.D. do IMPA

Rio de Janeiro
2020

AGRADECIMENTOS

Gostaríamos de agradecer a todos aqueles que contribuíram de forma direta ou indireta para a construção do nosso conhecimento durante toda a formação acadêmica.

Aos professores Paulo Rosa e Jauvane Cavalcante por terem nos acompanhado durante a longa jornada de elaboração desse trabalho, sendo bastante prestativos e dedicados ao nosso sucesso.

Aos professores Ronaldo Goldschmidt, Carla Pagliari e Julio Cesar Duarte, por terem contribuído com conselhos durante as avaliações parciais.

Por fim, a todos nossos amigos e familiares, que durante toda nossa jornada de graduação nos deram apoio emocional para que nos superássemos dia após dia.

RESUMO

Imersão em ambientes virtuais está se tornando mais popular. Diversas empresas, como *Samsung*, *Google* e *Facebook*, investem nestas tecnologias. Apesar disto, os produtos mais conhecidos, como o *Samsung Gear VR* e o *Google Card Board*, não oferecem nenhum tipo de suporte a interação natural, implementando interação por meio de botões. Já o *Oculus Rift*, do *Facebook*, oferece um meio de interação mais natural, usando as mãos, mas utiliza múltiplos sensores. Este trabalho descreve a criação de um sistema de baixo custo que permite a interação de um usuário com ambientes virtuais modelados em *Unity* e visualizados através de um celular em um *Google Cardboard* - um óculos de realidade virtual. O modelo proposto para interação se baseia na aplicação de algoritmos de *hand-tracking* em quadros capturados por uma única câmera RGB do celular. O objetivo é utilizar este sistema para realizar uma tarefa de *pick-and-place*. A contribuição deste trabalho consiste na criação e disponibilização do código, em um repositório *open-source*, do sistema proposto para integração em aplicações.

Palavras-chave: Redes Neurais. Aprendizado Profundo. Realidade Virtual. Sistema de Tempo Real. Análise de Imagens. Detecção de Pose.

ABSTRACT

Immersion in virtual environments is becoming more popular. Several companies, like *Samsung*, *Google* and *Facebook* have invested in these technologies. However, the well-known products, like the *Samgung Gear VR* and the *Google Card Board* offer no support for natural interaction with the virtual environment, but rather interaction through buttons. *Facebook's Rift Glasses* does offer a more natural interaction media, using hands, though it uses multiple sensors. The work describes the elaboration of a low-cost system that allows user interaction with VR environments modeled in Unity and viewed through a cellphone inside a Google Cardboard - virtual reality glasses. The model proposed for interaction is based on applying Hand-Tracking algorithms in frames captured by a single RGB camera of the cellphone. The main objective is to use this system to perform a *pick-and-place* task. The main contribution consists of creating and providing the proposed system's code in an open-source repository to integrate it into other applications easily.

Keywords: Neural Networks. Deep Learning. Virtual Reality. Real-time Systems. Image Analysis. Pose Tracking.

LISTA DE ILUSTRAÇÕES

Figura 1 – Utilização da rede YOLO (1) para reconhecimento de objetos em imagens	17
Figura 2 – Esquema mostrando o fluxo da transformação que ocorre na Cinemática Direta. Adaptado de Kucuk e Bingul(2)	20
Figura 3 – Exemplo de manipulador simples com juntas de um grau de liberdade.	21
Figura 4 – Esquema mostrando a ocorrência de <i>Gimbal Lock</i> em um sistema giroscópico. Imagem encontrada em Strickland(3).	21
Figura 5 – Esquema mostrando o fluxo da transformação que ocorre na Cinemática Inversa. Adaptado de Kucuk e Bingul(2)	22
Figura 6 – Saída do modelo criado em Cao et al.(4), com detecção dos pontos chave do corpo, dos pés, das mãos e do rosto. Imagem retirada de Cao et al.(4).	25
Figura 7 – Gráfico comparativo entre <i>OpenPose Mask</i> , <i>R-CNN</i> , e <i>Alpha-Pose</i> do tempo de execução em função do número de pessoas na imagem. Cada análise corresponde ao tempo médio de execução após 1000 execuções com uma placa de vídeo <i>Nvidia 1080 Ti</i> e <i>CUDA 8</i> . Imagem retirada de Cao et al.(4).	26
Figura 8 – No canto superior esquerdo a imagem usada, no quanto superior direito a posição e a articulação 3D estimados pelo modelo descrito em Panteleris, Oikonomidis e Argyros(5) e na parte de baixo a sobreposição do modelo na imagem original. Imagem retirada de Panteleris, Oikonomidis e Argyros(5)	27
Figura 9 – A primeira imagem é criada a partir de um modelo 3D, a segunda é a primeira imagem após ser tratada por uma rede neural e a terceira é a imagem gerada para compor grupo de testes para treino da rede neural de detecção da posição das mãos. Imagem retirada de Mueller et al.(6)	27
Figura 10 – Resultados do método de Mueller et al.(6). Imagem retirada de Mueller et al.(6)	28
Figura 11 – Esta figura representa os estágios do processamento de cada quadro, desde a captura no cliente e processamento no servidor até renderização em três dimensões no cliente.	29
Figura 12 – Imagem da mão usada como modelo à distância máxima da câmera do celular (posicionado a aproximadamente 5cm dos olhos). Nas dimensões originais da imagem, 640x480 pixels, o retângulo vermelho tem 260x260 pixels.	33
Figura 13 – Cena simples no <i>Unity</i> mostrando o anteparo com o quadro capturado junto dos pontos-chave e o esqueleto tridimensional após processamento no servidor.	34

Figura 14 – Cena simples no <i>Unity</i> mostrando a tela do celular, com a visão de cada olho separada. Nesta cena pode-se notar um objeto sendo segurado.	35
Figura 15 – Na esquerda imagem recebida do cliente e na direita imagem cortada na região de interesse - em torno da mão.	36
Figura 16 – Na esquerda imagem de entrada para rede neural e na direita imagem com pontos vermelhos em cima dos pontos detectados como de maior probabilidade para cada uma das 21 juntas.	37
Figura 17 – Modelo da mão utilizada neste projeto com eixos de rotação visíveis. Este modelo foi obtido do projeto de Panteleris, Oikonomidis e Argyros(5).	38
Figura 18 – Parte da sequência de quadros de teste, com o intervalo de 10 quadros a cada representação. Estão ordenados temporalmente da esquerda para a direita e de cima para baixo.	41
Figura 19 – Gráfico comparativo da saída do sistema da posição x do indicador, usando a rede neural de Mueller et al.(6) e a de Cao et al.(4).	42
Figura 20 – Gráfico comparativo da saída do sistema da posição x do indicador, com e sem o uso do algoritmo ótico de Lucas e Kanade(7).	43
Figura 21 – Gráfico comparativo da saída do sistema da posição x do indicador, com e sem o uso do filtro de Casiez, Roussel e Vogel(8).	44

LISTA DE TABELAS

Tabela 1 – Tempo médio de execução de tarefas realizadas no cliente e no servidor. Estes dados foram obtidos fazendo testes com duração de 200 quadros.	45
Tabela 2 – Tempo de execução médio para cada quadro, por tipo de conexão com e sem o uso de <i>VP8</i> , algoritmo de compressão de mídia em tempo real. Os dados locais foram obtidos usando os mesmos quadros de teste citados no inicio deste capítulo e os demais foram gerados a partir de testes com duração de 200 quadros.	46
Tabela 3 – Tempo médio de execução das tarefas do servidor durante o processamento de cada quadro.	46

LISTA DE ABREVIATURAS E SIGLAS

CFF	Critical Flicker Fusion
RGB	Red, Green and Blue
RGB-D	Red, Green, Blue and Depth
RV	Realidade Virtual

SUMÁRIO

1	INTRODUÇÃO	13
1.1	MOTIVAÇÃO	13
1.2	CARACTERIZAÇÃO DO PROBLEMA	14
1.3	OBJETIVO	14
1.4	JUSTIFICATIVA	14
1.5	METODOLOGIA	14
1.6	ESTRUTURA	15
2	FUNDAMENTAÇÃO TEÓRICA	17
2.1	RECONHECIMENTO DE OBJETOS	17
2.2	QUATÉRNIOS E ROTAÇÕES 3D	18
2.3	CINEMÁTICA DIRETA	20
2.4	CINEMÁTICA INVERSA	21
2.5	PROJEÇÃO DE PONTOS 3D	22
3	REVISÃO DE LITERATURA	24
3.1	MODELO PARA ESTIMAÇÃO DE POSE	24
3.2	SOLUÇÃO PARA ESTIMATIVA DE POSE 3D COM UMA CÂMERA COLORIDA	25
3.3	<i>GANERATED</i>	26
4	SOLUÇÃO PROPOSTA	29
4.1	CLIENTE	29
4.2	SERVIDOR	30
5	IMPLEMENTAÇÃO	32
5.1	CLIENTE	32
5.1.1	CAPTURA DE QUADROS, COMPRESSÃO DE QUADROS E ENVIO PARA O SERVIDOR	32
5.1.2	RECEBIMENTO DE DADOS E PROCESSAMENTO	33
5.1.3	ANALISE DE POSE DA GARRA	34
5.2	SERVIDOR	35
5.2.1	RECEBIMENTO E DESCOMPRESSÃO DE QUADROS	35
5.2.2	CORTE DA IMAGEM NA REGIÃO DE INTERESSE	35
5.2.3	DETECÇÃO DOS PONTOS CHAVE DA MÃO	36
5.2.4	APLICAÇÃO DE FILTRO NOS PONTOS DETECTADOS	37
5.2.5	MODELO DA MÃO	38

5.2.6	CINEMÁTICA DIRETA E PROJEÇÃO DOS PONTOS	39
5.2.7	CINEMÁTICA INVERSA	40
6	RESULTADOS E DISCUSSÕES	41
6.1	ANÁLISE COMPARATIVA DAS REDES NEURAIS	41
6.2	ANÁLISE DO DESEMPENHO DE ALGORITMOS PARA DIMINUIR RUÍDO	43
6.3	ANÁLISE DO TEMPO DE EXECUÇÃO DO SISTEMA INTEGRADO	44
6.4	ANÁLISE DO TEMPO DE EXECUÇÃO DO SERVIDOR	45
7	CONCLUSÃO	47
	REFERÊNCIAS	49

1 INTRODUÇÃO

Realidade Virtual (VR) pode ser entendido como a existência de uma interface capaz de causar a impressão a um usuário de que ele está presente em outro ambiente, em outra *realidade*, seja de forma completa ou parcial. Atualmente, esse conceito é dado a ambientes gerados por computadores, usando um conjunto de sensores que auxilie um sistema a reagir às movimentações e interações dos usuários (9).

Outro desafio bem conhecido na área de Visão Computacional é o reconhecimento e a estimativa da pose articulada do corpo humano e seus movimentos. O problema consiste em detectar de forma precisa e eficiente partes do corpo humano, bem como as articulações presentes, como cotovelos, joelhos, calcanhares, etc. O resultado da detecção é usualmente um conjunto de pontos (2D ou 3D) e as ligações entre eles, representando a pose estimada. As dificuldades que podem aparecer, além do desafio *per se* de se detectar partes distintas do corpo e juntá-las, são por exemplo: o corpo nem sempre estar completamente à mostra e alguns fatores externos como luminosidade e contraste da imagem que podem dificultar o processo. Apesar dessas barreiras, o reconhecimento de pose também se mostrou útil no cinema e em jogos, por exemplo. Além disso, as tecnologias nessa área têm melhorado bastante, criando ferramentas capazes de detectar as articulações do corpo sem a necessidade de muitos sensores auxiliares.

1.1 Motivação

O principal objetivo almejado durante a elaboração de um ambiente de Realidade Virtual (VR) é um alto grau de imersão, que pode ser definido como o quanto o usuário é capaz de sentir-se mentalmente imerso ou presente na simulação. Um exemplo simples é no caso da visão, em que pode ser usado um óculos que bloqueia a visão do mundo exterior, enquanto que as telas posicionadas em frente aos olhos do usuário apresentam os elementos visuais do ambiente. Outra maneira de aumentar o fator de imersão é melhorando a interatividade do usuário. Outros conceitos já foram relacionados com a imersão em ambientes virtuais, como o senso de presença e singularidade do ambiente (10).

Tendo em vista o alto empenho de desenvolvedores de experiências VR para criar um alto grau de imersão, percebe-se que ainda há muito a ser feito adicionando estímulos mistos aos sentidos do usuário. Com o uso de reconhecimento de pose para as mãos do usuário, mistura-se os estímulos sensoriais de visão e de tato. Com as tecnologias disponíveis no estado da arte para *hand-tracking*, o ferramental necessário já encontra-se disponível para incluir o *hand-tracking* em ambientes de VR.

1.2 Caracterização do Problema

O problema explorado neste trabalho é o de interagir com ambientes virtuais de forma natural em aplicações de VR por meio das mãos, sem haver um custo adicional elevado para o usuário final, ou seja, não exigir equipamentos específicos como câmeras RGB-D ou mesmo múltiplas câmeras e sim equipamentos comuns como um celular e um computador.

1.3 Objetivo

O objetivo final deste trabalho é desenvolver uma solução de baixo custo que viabilize a interação através das mãos em tempo real com ambientes de Realidade Virtual. Para isto, será desenvolvida uma interface capaz de receber quadros da câmera de um dispositivo que renderiza a experiência VR e processar esses quadros, fazendo o reconhecimento de pose tridimensional para as mãos do usuário e enviando para o dispositivo VR as informações dessa pose (a posição e rotação para cada grau de liberdade das 21 juntas presentes em uma mão), para que este possa renderizar um modelo 3D para as mãos do usuário. A fim de testar se objetivo foi alcançado, será verificada a capacidade de realizar uma atividade de *pick-and-place*, que consistirá em mover um cubo para uma área específica.

1.4 Justificativa

Espera-se que este trabalho aumente a disponibilidade de fatores de imersão ao disponibilizar o código de uma ferramenta *open-source* que permite Reconhecimento de Pose Tridimensional para mãos em ambiente virtual, usando apenas um celular e um servidor. Com o rastreamento das mãos do usuário e a possibilidade de interação com objetos em realidade virtual, torna-se desnecessário o uso de aparelhos auxiliares como sensores e controladores. Desta forma, fica a critério do desenvolvedor de analisar o *trade-off* entre custo e precisão aumentada por esses aparelhos.

1.5 Metodologia

Primeiro, foi feito um estudo do estado da arte de forma a compreender os principais desafios de problemas relacionados. A partir deste estudo inicial, foram escolhidos os trabalhos mais próximos do estado da arte, Panteleris, Oikonomidis e Argyros(5) e Mueller et al.(6), que resolviam o seguinte problema: a partir de imagens RGB de mãos, obter a pose tridimensional destas.

Para visualizar o ambiente virtual, optou-se pelo uso de um celular acoplado a um *Google Cardboard* - uma estrutura simples de papelão e duas lentes. Para a interação com

o ambiente, foram adaptadas as soluções estudadas.

O custo computacional das soluções propostas pelos trabalhos escolhidos é alto, o que poderia implicar em um tempo maior de processamento e um gasto elevado de bateria - além de maior complexidade de implementação das redes neurais em arquitetura *Android*. A fim de resolver este problema, o processamento foi dividido em uma arquitetura cliente-servidor. Nesta arquitetura, o cliente (celular) renderiza o mundo virtual para o usuário, faz todos os cálculos referentes a este e envia os quadros capturados pela câmera do celular para o servidor. Este recebe estes quadros, os processa usando uma rede neural para detectar os pontos-chave da mão e cinemática inversa para calcular a pose tridimensional. Por fim, o servidor envia dados referentes à esta pose para o cliente, o qual adiciona a mão ao ambiente virtual.

A fim de possibilitar a tarefa de *pick-and-place*, o cliente além de renderizar as mãos, também define regras e estados para segurar e soltar objetos.

Além disso, durante o andamento da pesquisa realizada nessa tese, o seguinte artigo foi aprovado para publicação:

- CASTRO, M. C. L. de; XAVIER, J. P. de A.; ROSA, P. F. F.; OLIVEIRA, J. C. de. Interação por Rastreamento de Mão em ambiente de Realidade Virtual. In: *22nd Symposium on Virtual and Augmented Reality (WUW-SVR2020)*. Porto de Galinhas, Brazil.

1.6 Estrutura

Primeiramente, os conceitos necessários para total compreensão do trabalho serão abordados no capítulo 2, Fundamentação Teórica. Estes conceitos são discutidos tanto em nível conceitual quanto em como cada um desses conhecimentos será aplicado no desenvolvimento da ferramenta.

No capítulo 3, Revisão de Literatura, as tecnologias já presentes no estado da arte para esse tipo de aplicação serão apresentados. Estes trabalhos serão mencionados e posteriormente citados em grande parte do decorrer deste relatório: os trabalhos desenvolvidos por Panteleris, Oikonomidis e Argyros(5), Cao et al.(4) e Mueller et al.(6).

Com isso, todo o acervo necessário para compreender o desenvolvimento deste trabalho terá sido abordado. No capítulo 4, Solução Proposta, o trabalho prossegue apresentando a arquitetura da solução. No capítulo 5, Implementação, é explicitada cada etapa do processo, abordando possíveis *trade-offs*, sendo a decisão dos caminhos tomados justificada adiante.

No capítulo 6, Resultados e Discussões, os resultados são expostos e discutidos. O

desempenho da ferramenta é analisado com diferentes configurações que foram consideradas na implementação, justificando assim as decisões tomadas no decorrer do trabalho.

Por fim, no capítulo 7, Conclusão, o conhecimento e os resultados obtidos são sintetizados. São apresentadas oportunidades de melhorias e ideias para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Primeiramente, é necessário expor e clarificar alguns conceitos fundamentais para a compreensão do problema e sua solução.

2.1 Reconhecimento de Objetos

O Reconhecimento de Objetos e padrões em imagens é uma área bastante frequente no âmbito científico. As aplicações são várias: Reconhecimento Facial (11), Veículos Autônomos (12) e detecção de doenças em seres vivos (13) são alguns exemplos.

O problema em geral consiste em, recebendo uma imagem como parâmetro de entrada, detectar um padrão ou algum objeto presente nela. Existem muitas implementações de soluções em casos específicos, como as soluções utilizando Redes Neurais Convolucionais (14), e algumas *frameworks* no estado da arte para esse tipo de problema, como a *YOLO* (1), que resolve o problema genérico com grande precisão e rapidez para vários tipos diferentes de objeto como na figura 1 abaixo, além de ter passado por otimizações com o tempo.

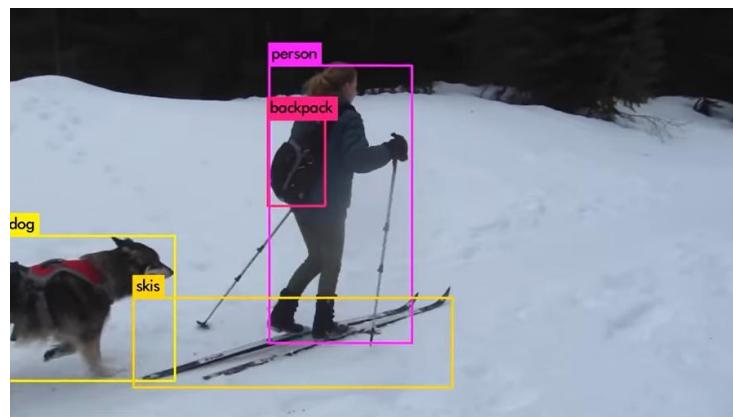


Figura 1 – Utilização da rede YOLO (1) para reconhecimento de objetos em imagens

Uma instância desse problema ocorre também aqui neste trabalho. Antes de se detectar cada junta presente na mão, é necessário localizar a mão como um todo na imagem. Será explicado mais adiante com mais detalhes como a *framework* YOLO (1) é utilizada neste caso.

2.2 Quatérnios e Rotações 3D

Os Quatérnios foram propostos como um novo sistema de números imaginários na álgebra. A princípio o entendimento dos Quatérnios era usado para quocientes de segmentos de reta 3D por Hamilton (15), mas hoje em dia seu escopo foi bastante ampliado e é utilizado inclusive em outras áreas além da geométrica, como a Mecânica Quântica (16).

Para a síntese do conhecimento neste tópico, serão usadas como base as abordagens utilizadas em Schmidt e Niemann(17) e Sanderson e Eater(18). O conjunto dos Quatérnios é um conjunto de números (normalmente representado por H), e seus elementos são definidos como:

$$h = w + xi + yj + zk \quad (2.1)$$

Sendo $w, x, y, z \in R$. i, j e k são unidades imaginárias que satisfazem as seguintes propriedades:

$$\begin{aligned} i^2 &= j^2 = k^2 = -1, \\ ij &= -ji = k, \\ jk &= -kj = i, \\ ki &= -ik = j \end{aligned} \quad (2.2)$$

Dessa forma, os Quatérnios também podem ser definidos como um par $h = (w, \vec{v})$, sendo w um número real e \vec{v} um vetor tridimensional ($\vec{v} = x\hat{i} + y\hat{j} + z\hat{k}$). Outra forma comum de se representar esses números é na sua forma polar (também chamada de exponencial ou forma ângulo-eixo):

$$h = H(\cos\theta, \vec{I}\sin\theta) \quad (2.3)$$

Sendo

$$\begin{aligned} H &= \sqrt{w^2 + x^2 + y^2 + z^2}, \\ \theta &= \arccos(w/H), \\ \vec{I} &= \frac{\vec{v}}{|\vec{v}|} \end{aligned} \quad (2.4)$$

O produto de dois Quaternios é feito aplicando a propriedade distributiva utilizando as regras definidas por 2.2. Utilizando a notação vetorial, tem-se o seguinte resultado:

$$h_1 \cdot h_2 = (w_1 w_2 - \vec{v}_1 \cdot \vec{v}_2, w_1 \vec{v}_1 + w_2 \vec{v}_2 + \vec{v}_1 \times \vec{v}_2) \quad (2.5)$$

Sendo $h_1 = (w_1, \vec{v}_1)$, $h_2 = (w_2, \vec{v}_2)$, e os operadores \cdot e \times definidos entre os vetores \vec{v}_1 e \vec{v}_2 sendo o produto escalar e vetorial, respectivamente.

É importante ressaltar que, pelo resultado obtido, observa-se que o produto de dois quatérnios não é comutativo.

Algumas definições extras com relação aos quatérnios podem ser dadas após a definição de seu produto: Seja $\bar{h} = (w, -\vec{v})$ o *conjugado* de um quatérnio. A *norma* ou *módulo* de um quatérnio, $|h|$, é dada então por $|h| = \sqrt{h \cdot \bar{h}} = \sqrt{w^2 + x^2 + y^2 + z^2}$. O *inverso* de um quatérnio, h^{-1} , é definido como um número tal que $h \cdot h^{-1} = 1$. Perceba que, se $|h| = 1$, então $h^{-1} = \bar{h}$.

A utilidade dos Quatérnios é dada por exemplo para calcular rotações de pontos tridimensionais num computador de forma eficiente. Além disso, a utilização de quatérnios nesse tipo de problema retira algumas ambiguidades na representação de rotações com relação a outras representações como Ângulos de Euler, por exemplo, o que leva a alguns problemas como o *Gimbal Lock*.

Seja \vec{p} um ponto representado num ambiente tridimensional. O problema da rotação consiste, então, em obter o ponto \vec{q} resultante da rotação de \vec{p} em torno de um eixo \vec{v} unitário num valor de θ radianos no sentido anti-horário. Essa transformação é feita através da seguinte relação:

$$Q = H \cdot P \cdot H^{-1} \quad (2.6)$$

Sendo

$$\begin{aligned} H &= (\cos \frac{\theta}{2}, \vec{v} \operatorname{sen} \frac{\theta}{2}) \\ P &= (0, \vec{p}) \end{aligned} \quad (2.7)$$

Note que $|H| = 1$ pois \vec{v} é unitário, então $H^{-1} = \bar{H}$. O resultado dessa operação é $Q = (0, \vec{q})$, pelo qual se obtém o ponto \vec{p} rotacionado.

O conceito das operações em quatérnios será importante mais adiante para a compreensão de como o computador fará as operações de rotação em torno do eixo das juntas em um problema de cinemática direta. No entanto, em todos os momentos que as

operações com quatérnios forem mencionadas, haverá uma breve explicação do significado por trás destas operações.

2.3 Cinemática Direta

Os problemas da Cinemática Direta e Cinemática Inversa são subcampos da Robótica relacionados ao estudo do movimento de manipuladores robóticos. A transformação que ocorre na Cinemática Direta é, essencialmente, uma transformação que vai dos parâmetros das juntas de um manipulador robótico (mais comumente os ângulos em cada junta) para as posições das juntas no espaço. A imagem 2 adaptada de Kucuk e Bingul(2) ilustra o fluxo de atuação da Cinemática Direta.

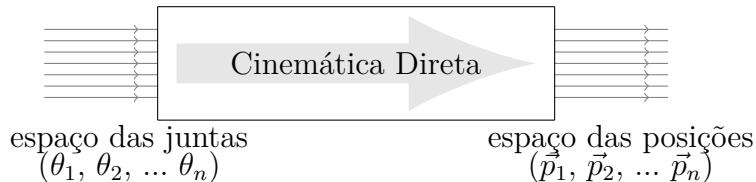


Figura 2 – Esquema mostrando o fluxo da transformação que ocorre na Cinemática Direta.
Adaptado de Kucuk e Bingul(2)

No escopo desse trabalho, trabalha-se na maioria das vezes com manipuladores com juntas que possuem apenas um grau de liberdade. Nesse caso, seja $\theta_1, \theta_2, \dots, \theta_n$ os ângulos de rotação de cada junta relativo ao modelo original em torno de seu eixo de rotação. Nesse caso, tem-se que:

$$\vec{p}_i = \vec{p}_{i-1} + Q(\theta_i) \cdot S_i \cdot Q(\theta_i)^{-1} \quad (2.8)$$

Onde:

- \vec{p}_i o vetor posição da junta i com relação ao espaço;
- $Q(\theta_i)$ o quatérnio $(\frac{\theta}{2}, \vec{u})$, com \vec{u} o eixo de rotação da junta i ;
- θ_i o angulo de rotação aplicada na junta;
- S_i o vetor posição relativa original da junta i com a junta $i - 1$.

Um esquema ilustrativo é dado pela figura 3 a seguir, ilustrando o caso para 3 juntas, sendo as juntas 1 e 2 com um grau de liberdade.

Note que esse esquema é similar ao formato de como os dedos de uma mão se comportam. Mais adiante será apresentado que, das 21 juntas de um modelo tridimensional

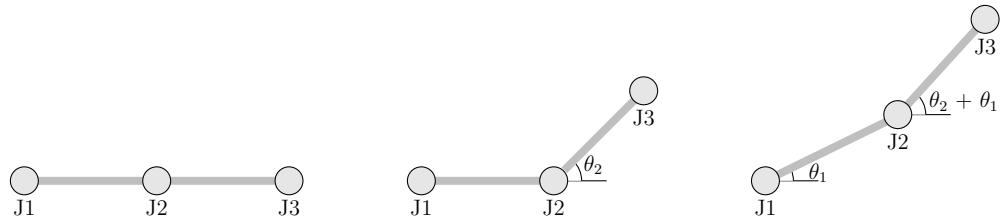


Figura 3 – Exemplo de manipulador simples com juntas de um grau de liberdade.

de uma mão, apenas 6 têm dois ou mais graus de liberdade, e o pulso contém três graus de liberdade relativos à rotação.

O raciocínio para dois graus de liberdade é análogo: Quando há dois graus de liberdade a rotação pode ser feita da mesma forma, rotacionando em torno dos dois eixos que compõem a junta. No caso de três graus de liberdade, a rotação deve ser feita com cuidado: Isso se dá devido à ocorrência de *Gimbal Lock*. Esse fenômeno ocorre quando dois ou mais eixos de rotação coincidem em uma determinada configuração como na figura 4, ocorrendo a perca de um dos graus de liberdade. Nesse caso é ideal utilizar a rotação parametrizando diretamente por Quaternios ao invés de ângulos.

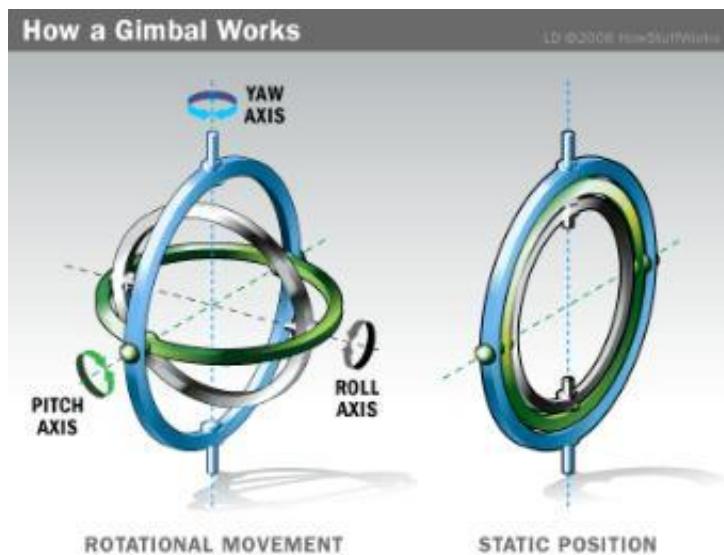


Figura 4 – Esquema mostrando a ocorrência de *Gimbal Lock* em um sistema giroscópico.
Imagen encontrada em Strickland(3).

2.4 Cinemática Inversa

O problema da Cinemática Inversa é, como o nome sugere, o processo inverso da Cinemática Direta. Isto é, dadas as posições desejadas em que cada junta esteja, determinar a configuração de rotação de cada junta para alcançar estas posições. Note que este é um problema bem mais custoso computacionalmente e nem sempre tem solução exata,

diferente da Cinemática Direta. A imagem 5 ilustra o fluxo de atuação da Cinemática Inversa.

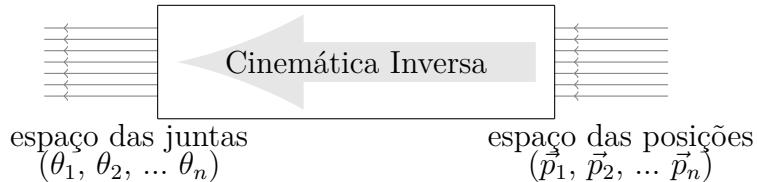


Figura 5 – Esquema mostrando o fluxo da transformação que ocorre na Cinemática Inversa.
Adaptado de Kucuk e Bingul(2)

Outra observação a ser feita é que na Cinemática Direta a solução segue um algoritmo pelo qual é possível obter a solução analítica para o problema. Já no caso da Cinemática Inversa essa solução analítica pode não existir. Quando ela não existe, utiliza-se métodos numéricos para encontrar a configuração mais apropriada.

A importância da Cinemática Inversa se dá, por exemplo, na essência dos Manipuladores Robóticos: Enquanto normalmente as tarefas dos manipuladores é feita no ambiente Cartesiano (posições), os atuadores realizam o controle através dos parâmetros (juntas).

No problema do modelo tridimensional de uma mão aqui realizado, utiliza-se a transformação de projeção 2D dos pontos de cada junta e então é aplicado o método numérico de Moré(19) para otimizar o modelo. Esse problema é resolvido por Panteleris, Oikonomidis e Argyros(5) como um problema de Cinemática Inversa.

2.5 Projeção de pontos 3D

Neste tópico será abordado como pontos tridimensionais podem ser projetados em um anteparo, dados os parâmetros da câmera e as posições desses pontos e do anteparo.

A transformação sem considerar distorções da câmera segue o proposto por Calibration(20):

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2.9)$$

Sendo:

- x, y e z são as coordenadas do ponto;

- u e v são as coordenadas xy dos pontos no anteparo;
- f_x , f_y , c_x e c_y são os parâmetros intrínsecos da câmera;
- A matriz restante, chamada de $[R|T]$ é a matriz com os parâmetros extrínsecos da câmera (as matrizes de rotação e translação)

Essa transformação, como já afirmado, desconsidera os fatores de distorção que ocorrem na câmera. Quando esta ocorre, as transformações não seguem mais a linearidade, e a extensão do modelo, proposto por Calibration(20), é da seguinte forma:

$$\begin{aligned} x' &= x/z \\ y' &= y/z \\ x'' &= Kx' + 2p_1x'y' + p_2(r^2 + 2x'^2) \\ y'' &= Ky' + p_1(r^2 + 2y'^2) + 2p_2x'y' \end{aligned} \tag{2.10}$$

Sendo:

- $K = (1 + k_1r^2 + k_2r^4 + k_3r^6)/(1 + k_4r^2 + k_5r^4 + k_6r^6)$
- $r^2 = (x')^2 + (y')^2$
- k_1 , k_2 , k_3 , k_4 , k_5 e k_6 os fatores de distorção radial da câmera, e p_1 e p_2 fatores de distorção tangencial.

Esses parâmetros intrínsecos, como distância focal e os fatores de distorção, podem ser computados após uma etapa de calibração. A câmera éposta para obter várias vistas de um padrão determinado, e assim a partir da informação coletada na imagem os valores dos parâmetros são encontrados.

3 REVISÃO DE LITERATURA

O problema de detecção de partes do corpo humano é um desafio para o qual estão sendo desenvolvidas diversas soluções com aproximações variadas. Algumas destas serão discutidas neste capítulo. Elas foram escolhidas com base nos requerimentos de hardware, na eficiência e na capacidade de executar as tarefas propostas.

Os equipamentos utilizados estão sendo levados em consideração com base no objetivo deste trabalho, principalmente o fato de que apenas um celular deve ser utilizado para a captura dos quadros. Soluções baseadas em câmeras com sensor de profundidade (RGB-D) não serão consideradas, visto que celulares não costumam ter este tipo de câmera. Além disso, este sensor tem pouca eficiência ao ar livre, por conta da intensidade da luz do sol. Câmeras Térmicas também podem ser usadas para aumentar a precisão da estimativa de pose e são menos dependentes de luminosidade (21), mas soluções que dependam deste tipo de câmera também não serão consideradas visto que não são comuns em celulares atualmente.

Também é necessário considerar que a interação deve ser feita em tempo real ou próximo, a fim de obter imersão no mundo virtual. Para isto, a taxa de quadros deve ser maior que a *Critical Flicker Fusion Threshold* (CFF), um valor que varia de pessoa pra pessoa e também por fatores externos ao corpo humano, como descrito em Eisen-Enosh et al.(22) e Tyler(23). No entanto, com a capacidade de processamento atualmente disponível, essa taxa não é alcançada facilmente.

3.1 Modelo para estimativa de pose

O *OpenPose* (4) é um sistema capaz de resolver o problema do Reconhecimento de Pose (isto é, determinar a partir de parâmetros de entrada como imagem e vídeo). Ele consegue estender o problema para estimar a posição de múltiplas pessoas - posição de cada junta dos múltiplos esqueletos - em tempo real, sendo o estado da arte para este problema. Uma rede neural é utilizada para detectar pontos chave do corpo (por exemplo joelho esquerdo e o pulso direito) e, em seguida, é utilizado um algoritmo de correspondência em grafo bipartido. Um exemplo de saída do *OpenPose* pode ser observado na figura 6.

Este algoritmo de correspondência tenta ligar as partes do corpo para criar o esqueleto, ligando por exemplo um pulso direito em um cotovelo direito. Para múltiplas pessoas na mesma imagem, as partes são detectadas da mesma forma e entre todos os pares que poderiam ser ligadas cria-se uma aresta cujo peso depende da uniformidade da imagem entre os dois pontos. Depois disto, aplica-se um algoritmo para escolher as arestas

de forma a maximizar o peso total. Desta forma, apesar da complexidade do algoritmo ser polinomial em relação ao número de pessoas na foto, a parte executada na rede neural, a qual tem complexidade constante em relação ao número de pessoas, tem uma constante muito mais alta, de forma que o tempo de execução não aumenta significativamente com o número de pessoas na imagem. O gráfico da figura 7 compara o tempo de execução em função do número de pessoas a outras abordagens.

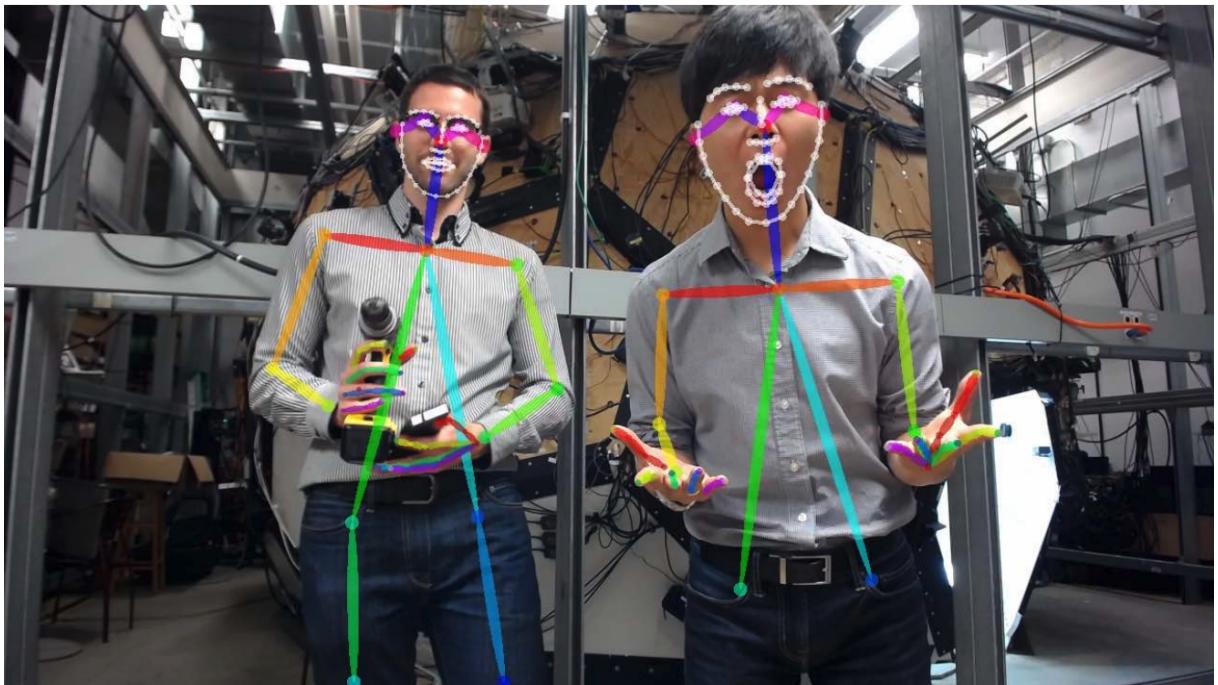


Figura 6 – Saída do modelo criado em Cao et al.(4), com detecção dos pontos chave do corpo, dos pés, das mãos e do rosto. Imagem retirada de Cao et al.(4).

A solução apresentada em Cao et al.(4) para estimação de pontos em três dimensões envolve usar múltiplas câmeras, o que vai de encontro à restrição de usar apenas um celular comum. Apesar disto, o modelo criado para obter os pontos-chave da mão em duas dimensões a partir de uma imagem pode ser utilizado junto de outras soluções para obter os pontos tridimensionais.

3.2 Solução para estimativa de pose 3d com uma câmera colorida

Conforme descrito, o maior problema da abordagem anterior é a necessidade de mais de uma câmera o que pode ser contornado com a abordagem em Panteleris, Oikonomidis e Argyros(5).

A abordagem discutida em Panteleris, Oikonomidis e Argyros(5) usa as saídas da rede neural de Cao et al.(4) para uma câmera colorida comum, ou seja, os pontos das articulações em duas dimensões. A partir destes pontos é calculada a pose tridimensional de um modelo cinematográfico de uma mão, tal que a distância entre os pontos detectados pelo

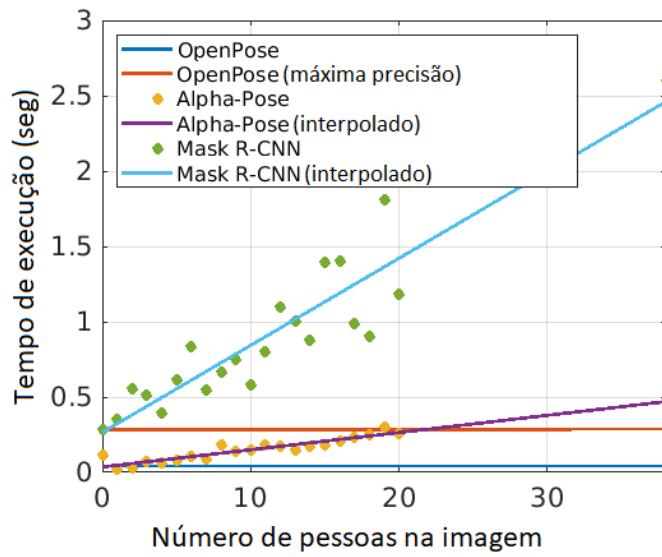


Figura 7 – Gráfico comparativo entre *OpenPose Mask*, *R-CNN*, e *Alpha-Pose* do tempo de execução em função do número de pessoas na imagem. Cada análise corresponde ao tempo médio de execução após 1000 execuções com uma placa de vídeo *Nvidia 1080 Ti* e *CUDA 8*. Imagem retirada de Cao et al.(4).

OpenPose e os pontos da projeção deste modelo sejam minimizados. Com a posição do modelo e o tamanho do esqueleto, a informação da pose aproximada é extraída usando métodos de regressão não linear. A figura 8 ilustra a saída deste modelo.

Esta abordagem atinge em torno de 24 milímetros de erro em 80% dos quadros em cada coordenada, um total de 41 milímetros no módulo da distância no vetor de erro em 80% dos casos. O tempo de processamento alcançou 18 quadros por segundo com uma placa de vídeo *NVIDIA GTX 1070* (5).

3.3 *GANerated*

A abordagem feita em Mueller et al.(6) consegue estimar a posição das mãos usando uma rede neural que tem como saída a posição dos pontos-chave em relação ao pulso em três dimensões e em duas dimensões em relação à imagem. Esta rede é rápida o suficiente para suportar processamento em tempo real e tem como grande vantagem precisar apenas de uma câmera colorida comum (RGB). A grande diferença em relação a outras abordagens é o grupo de teste utilizado para treinar. O grupo foi gerado a partir de modelos artificiais, a fim de obter alta precisão na posição das juntas e gerar um grande conjunto de teste. Os modelos renderizados passam por uma rede neural para aumentar a semelhança com a realidade, a fim de melhorar os resultados após o treinamento, e finalmente são adicionados diversos fundos, conforme visto na figura 9.

Para descobrir o valor de variáveis globais de posição, rotação e pose da mão, uma



Figura 8 – No canto superior esquerdo a imagem usada, no quanto superior direito a posição e a articulação 3D estimados pelo modelo descrito em Panteleris, Oikonomidis e Argyros(5) e na parte de baixo a sobreposição do modelo na imagem original. Imagem retirada de Panteleris, Oikonomidis e Argyros(5)



Figura 9 – A primeira imagem é criada a partir de um modelo 3D, a segunda é a primeira imagem após ser tratada por uma rede neural e a terceira é a imagem gerada para compor grupo de testes para treino da rede neural de detecção da posição das mãos. Imagem retirada de Mueller et al.(6)

medida de energia foi criada de tal forma que os valores que minimizam esta energia se aproximam do valor das variáveis utilizadas no modelo 3d renderizado. O primeiro termo considera a diferença da posição dos pontos-chave projetados na imagem e a posição

detectada pela rede neural em duas dimensões - para cada ponto chave, é somado quadrado da distância entre os pontos. O segundo termo é similar, considerando os pontos em três dimensões, ao invés de dois. O terceiro termo leva em consideração a articulação e os limites físicos, somando o quadrado do módulo do vetor do ângulo sólido que estiver além do limite para cada articulação. O último termo é referente à suavidade temporal, o quadrado do módulo do vetor de movimento, em relação à última medida, é adicionado à energia.

Esta abordagem permite inferir a posição em três dimensões das mãos, interpolando os resultados encontrados para a posição em duas dimensões (que contém informação sobre a posição em relação à câmera), a posição das articulações em três dimensões em relação ao pulso e o tamanho da mão na imagem em relação ao tamanho real. O resultado deste método pode ser observado na figura 10.



Figura 10 – Resultados do método de Mueller et al.(6). Imagem retirada de Mueller et al.(6)

O foco nos diferentes aspectos do treinamento da rede neural nesta abordagem proporciona um erro de 25 milímetros em mais de 90% dos pontos chave de articulação, um resultado superior a outras abordagens no estado da arte, Zimmermann e Brox(24) e Zhang et al.(25).

Este modelo falha em casos nos quais a cor de fundo da imagem coincide com a cor das mãos e quando a mão aparece apenas parcialmente na imagem. Apesar disto, para o objetivo deste trabalho ambos efeitos podem ser controlados sem perdas nos resultados.

A grande vantagem do uso desta rede é o fato de poder depender apenas de uma câmera colorida comum, em comparação com Cao et al.(4), e ao mesmo tempo obter uma precisão alta o suficiente para manter a sensação de imersão no ambiente virtual.

4 SOLUÇÃO PROPOSTA

O atual problema é permitir interação das mãos em um ambiente virtual em tempo real, usando um celular comum tanto para capturar quadros quanto para reproduzir o ambiente virtual.

A solução proposta para este problema consiste em dividir a arquitetura em cliente-servidor, mantendo o processamento mais custoso do lado do servidor, a fim de diminuir o tempo total de processamento e atingir processamento em tempo real. A figura 11 ilustra a arquitetura proposta, evidenciando ainda os processos que serão realizados desde a captura de quadro até a renderização das mãos e a interação com o ambiente.

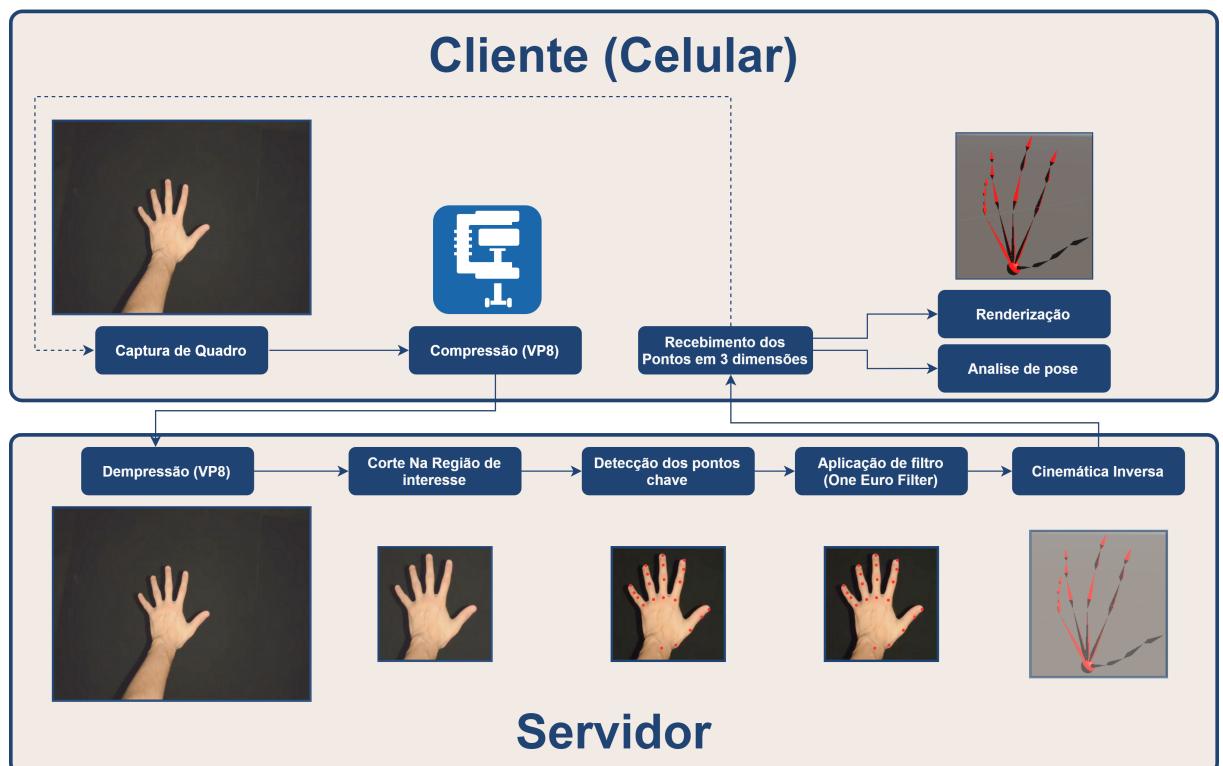


Figura 11 – Esta figura representa os estágios do processamento de cada quadro, desde a captura no cliente e processamento no servidor até renderização em três dimensões no cliente.

4.1 Cliente

Tendo em vista a arquitetura apresentada, o processamento do lado do celular (isto é, o cliente) consiste em:

1. Capturar a imagem da Câmera;

2. Comprimir a imagem e enviar para o servidor;
3. Receber de volta os pontos-chave da mão em três dimensões;
4. Renderizar um modelo da mão no ambiente virtual;
5. Fazer processamentos menores referentes a interação com o ambiente virtual.

No inicio de cada processamento, um quadro - idealmente com uma mão - vai ser capturado pela câmera do celular inserido no *Google Cardboard*. Após a captura, este quadro deve ser enviado para o servidor. A fim de diminuir os atrasos de transferência e consequentemente os atrasos totais, há uma etapa adicional de compressão de *stream* de vídeo antes do envio - desta forma, parte das redundâncias temporais e espaciais são removidas, implicando em uma quantidade menor de dados.

Após o envio do quadro, e subsequente processamento do servidor, o cliente deve receber a pose da mão em 3 dimensões, caso houvesse uma mão no quadro capturado. A pose descrita se refere à posição de 21 pontos-chave, um para cada junta, mais os ângulos de rotação de cada articulação e a rotação do pulso.

Com dados da pose, o cliente é capaz de renderizar a mão no ambiente virtual. Além disto deve haver um passo que calcula interações com o ambiente.

4.2 Servidor

Do lado do servidor, o processamento consiste em:

1. Aguardar o cliente enviar cada quadro;
2. Receber e descomprimir o quadro;
3. Detectar os pontos-chave em duas dimensões usando redes neurais conhecidas;
4. Realizar cinemática inversa para obter os pontos-chave em três dimensões;
5. Responder o cliente com os pontos-chave em três dimensões.

O servidor inicialmente aguarda o envio de um quadro pelo cliente. Após receber o quadro há um passo de descompressão, a fim de recuperar o quadro original.

Após isto, o quadro é recortado na região de interesse, destacando a mão do resto da imagem como preprocessamento para passar pelo algoritmo que irá detectar os 21 pontos-chave da mão em duas dimensões, na etapa seguinte. Após a detecção, há um passo adicional de Aplicação de Filtro, adicionado com a finalidade de diminuir ruído na posição das pontos-chave em função do tempo.

A fim de recuperar a informação completa da pose da mão em três dimensões a partir da projeção dos pontos, assim como em Panteleris, Oikonomidis e Argyros(5), modela-se um problema de cinemática inversa, que tenta achar a melhor pose cujos pontos projetados no plano do quadro estejam o mais próximo possível do detectado pela rede.

Finalmente, a informação da pose é retornada para o cliente.

5 IMPLEMENTAÇÃO

As próximas sessões irão explicitar todo o processo pelo qual cada quadro passa, conforme ilustrado pela figura 11. Vale observar que sempre que o cliente recebe os pontos em três dimensões, um novo quadro é capturado e enviado ao servidor, de tal forma que qualquer mudança de pose na mão irá ser refletida no ambiente virtual - o atraso entre a ação no mundo real e a reação no mundo virtual será discutido no capítulo de Resultados e Discussões.

Essa abordagem permite que o celular utilizado não fique sobrecarregado, utilizando-o apenas para interface e renderização do VR, que são essencialmente tarefas atribuídas a este aparelho nesta aplicação. Ao diminuir a carga de processamento do celular, diminui-se o consumo de bateria causado por essa aplicação, bem como aumenta a velocidade de processamento por estar utilizando o processador do computador. É válido também ressaltar que a taxa de quadros processados e renderizados por segundo também terá que considerar a transferência de dados entre o celular e o computador, que pode se tornar um gargalo no processo. Isso será discutido com mais detalhes na área de Resultados e Discussões.

5.1 Cliente

O cliente foi implementado usando a ferramenta de desenvolvimento de jogos *Unity* (26), com o intuito de reduzir a complexidade do projeto na parte de visão computacional, visto que esta ferramenta fornece abstrações para objetos, câmeras e luzes, além de outros elementos. Outro motivo é o fato desta ferramenta ser bem documentada e popular atualmente segundo Mishra e Shrawankar(27), desta forma desenvolvedores podem integrar esse processamento na criação de novas aplicações.

A integração do cliente com o servidor foi implementada de forma modular, e a implementação do cliente é simples. Desta forma outras aplicações podem se integrar ao servidor, enviando quadros e recebendo os ângulos e pontos em três dimensões referentes à pose da mão e realizar o próprio processamento para renderização da pose ou para outras finalidades.

5.1.1 Captura de quadros, compressão de quadros e envio para o servidor

O inicio do processamento de cada quadro corre na captura deste usando uma das câmeras do celular. A API do *Unity* foi usada para a captura de quadros de 640x480 pixels no formato RGB.

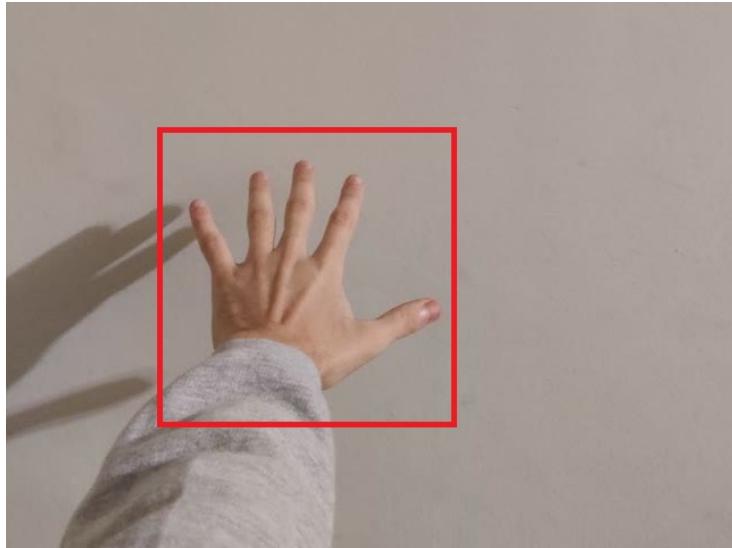


Figura 12 – Imagem da mão usada como modelo à distância máxima da câmera do celular (posicionado a aproximadamente 5cm dos olhos). Nas dimensões originais da imagem, 640x480 pixels, o retângulo vermelho tem 260x260 pixels.

As dimensões dos quadros foram escolhidas de tal forma que as dimensões da região da mão à distância máxima da câmera, a qual será recortada pelo servidor, tenha dimensões próximas à entrada da rede neural utilizada para processar o recorte no servidor, conforme ilustrado pela imagem 12. Desta forma não há perda de informação e o tempo para a transferência do quadro - além outros processos cuja complexidade depende das dimensões do quadro - é minimizado.

Após a captura da imagem, foi implementado um passo adicional de compressão usando o algoritmo *VP8*, criado pela *Google* (28) - outros formatos como *AV1* (29) não foram testados. O passo de compressão foi escrito em C++ usando as bibliotecas fornecidas pela *Google*, compilado para a arquitetura *Android* e adicionado ao *Unity* como código nativo.

Finalmente o quadro comprimido é enviado ao servidor para processamento.

5.1.2 Re却bimento de dados e processamento

Após o processamento, o servidor envia de volta para o cliente os ângulos das juntas, a posição do pulso relativo à câmera, a rotação do pulso e adicionalmente todos os pontos-chave da mão em três e duas dimensões.

Um objeto criado para manter estes dados é atualizado e chama todos os métodos registrados para alertar a atualização dos dados. A implementação atual chama três métodos: o primeiro método atualiza o anteparo mostrado na imagem 13; o segundo destrói o esqueleto anterior e recria baseado nos pontos-chave, usando uma esfera para o pulso e octaedros para ligar os pontos; o terceiro analisa a distância entre a ponta do indicador e

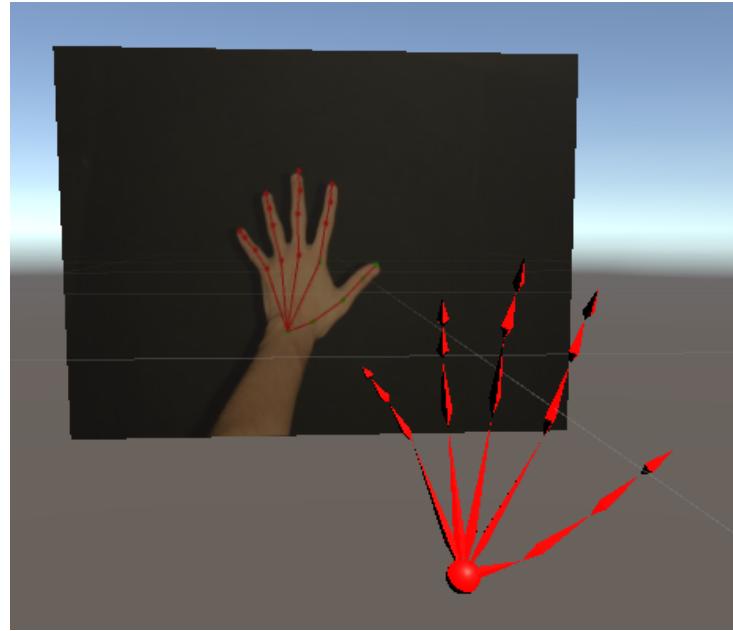


Figura 13 – Cena simples no *Unity* mostrando o anteparo com o quadro capturado junto dos pontos-chave e o esqueleto tridimensional após processamento no servidor.

a ponta do polegar para definir se está ou não tentando agarrar um objeto.

5.1.3 Analise de pose da garra

Para analisar se a mão está segurando um objeto ou não utiliza-se uma máquina de estados tendo os estados “Garra Fechada” e “Garra Aberta”. A cada atualização nos dados, o estado é atualizado baseado nas seguinte condições:

- Se o estado atual é “Garra Aberta” e a distância entre a ponta do indicador e a ponta do polegar é maior que um determinado limite, o próximo estado é “Garra Aberta”.
- Se o estado Atual é “Garra Aberta” e a distância é menor que o limite, o próximo estado é “Garra Fechada” e o objeto segurável mais próximo dentro de outro limite passa a ter o estado “Sendo Segurado”.
- Se estado atual é “Garra Fechada” e a distância é menor que o limite, o próximo estado é “Garra Fechada”, nada muda no estado de outros objetos.
- Se o estado atual é “Garra Fechada” e a distância é maior que o limite, o objeto que estava no estado “Sendo Segurado” passam para o estado “Solto”.

Enquanto um objeto está no estado “Sendo Segurado”, a cada atualização a variação na posição do ponto médio entre o indicador e o polegar é somado à posição do objeto. A

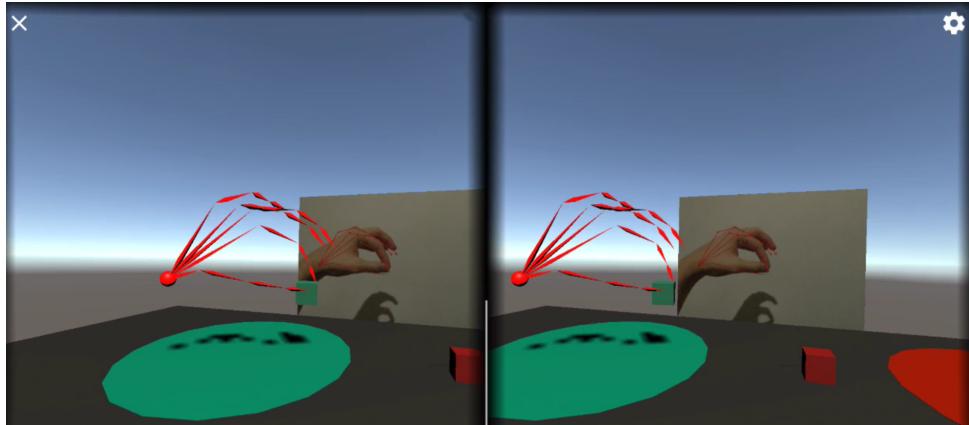


Figura 14 – Cena simples no *Unity* mostrando a tela do celular, com a visão de cada olho separada. Nesta cena pode-se notar um objeto sendo segurado.

imagem 14 demonstra o estado “Garra Fechada” enquanto o cubo verde é um objeto no estado “Sendo Segurado”.

5.2 Servidor

Conforme dito, a maior parte do processamento está no servidor. Este recebe cada quadro, descomprime, recorta, encontra os pontos-chave, faz a cinemática inversa baseado nestes pontos e retorna a pose da mão em três dimensões para o cliente, conforme visto na imagem 11.

A implementação foi feita em C++ tendo em vista otimizações no desempenho e a disponibilidade de bibliotecas de alto desempenho de visão computacional, álgebra linear e de regressão não-linear.

5.2.1 Recebimento e descompressão de quadros

O servidor fica ouvindo uma determinada porta para receber as solicitações de processamento de um quadro. Esse processamento não é feito de forma funcional como a descompressão depende dos quadros recebidos anteriormente e, conforme será visto, a pose é inicializada com a pose do quadro anterior.

Quando uma nova solicitação chega, os quadros comprimidos são recebidos e decodificados com *VP8*, a mesma biblioteca (*VPX*) utilizada no cliente. Usando a biblioteca *OpenCV* os quadros são então convertidos de *YV12* para *GBR*.

5.2.2 Corte da imagem na região de interesse

A tarefa de detectar os pontos-chave da mão é complexa e computacionalmente custosa. O tempo de execução da rede tem uma relação direta com o tamanho da imagem

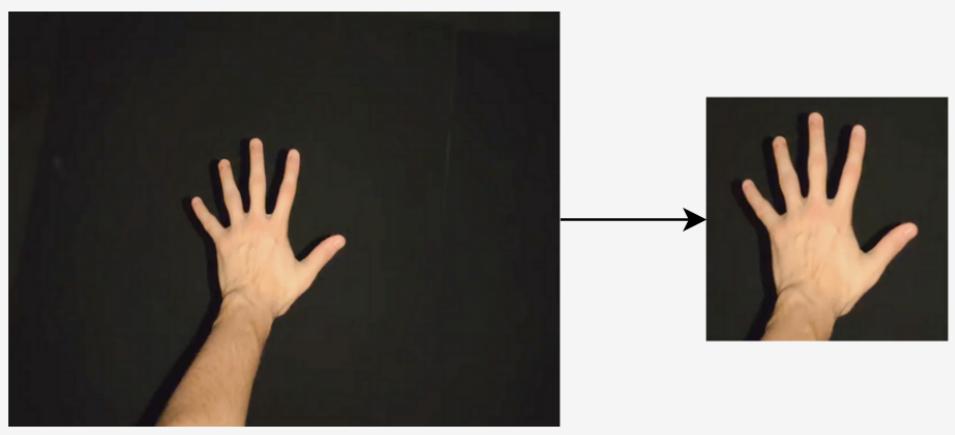


Figura 15 – Na esquerda imagem recebida do cliente e na direita imagem cortada na região de interesse - em torno da mão.

e, em contrapartida, imagens de tamanhos menores contém menos informação e diminuem a precisão do resultado. Como o objetivo é apenas detectar os pontos-chave da mão, a parte da imagem que contém a mão pode ser detectado e recortado do restante primeiro. Na implementação deste trabalho há apenas o recorte de um quadrado cujos lados são paralelos aos eixos da imagem como ilustrado na imagem 15.

A rede neural descrita em Redmon e Farhadi(1) foi criada para detectar a posição de objetos em imagens para qual foi treinada. No presente trabalho, uma rede deste tipo, que foi treinada para detectar mãos, foi utilizada para determinar a região de interesse da imagem. Apesar de resolver o problema inicial, usar esta rede em todos os quadros aumenta o tempo médio de execução, em vez disto, ela só é utilizada no primeiro quadro ou quando a detecção dos pontos-chave tem baixa confiança, o que pode indicar que a região recortada não contém mais os pontos. Nos demais quadros, é recortada uma região que é calculada baseada na posição dos pontos detectados no quadro anterior.

Vale notar que movimentos bruscos podem fazer o tempo de processamento de cada quadro aumentar, uma vez que será necessário utilizar mais vezes a rede neural para detecção da região de interesse. Baixa quantidade de quadros processados por segundo também pode ter o mesmo efeito.

5.2.3 Detecção dos pontos chave da mão

Para fazer a detecção dos pontos chave, basta redimensionar a região de interesse para o tamanho esperado pela entrada da rede neural. As redes neurais de Cao et al.(4) e Mueller et al.(6) foram testadas e os resultados serão discutidos - apesar da rede de Mueller et al.(6) também retornar os pontos em três dimensões em relação ao pulso, apenas os pontos em duas dimensões foram usados para fazer a regressão não linear no passo de cinemática inversa.

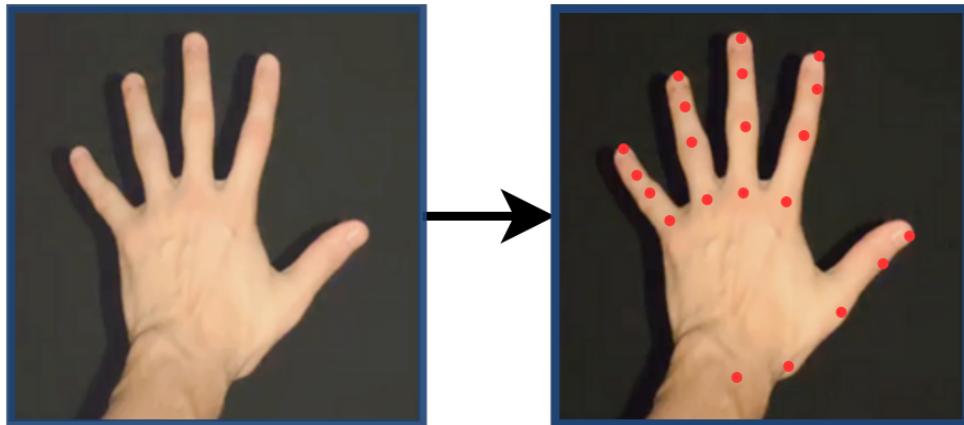


Figura 16 – Na esquerda imagem de entrada para rede neural e na direita imagem com pontos vermelhos em cima dos pontos detectados como de maior probabilidade para cada uma das 21 juntas.

A saída das duas redes neurais são 21 matrizes bidimensionais cujo valor em cada posição na i -ésima matriz é a probabilidade da i -ésima junta estar nesta posição, após redimensionamento, na imagem original. Apenas a informação do ponto com probabilidade máxima e o valor desta probabilidade, a confiança, são utilizados nos próximos passos. A figura 16 ilustra uma possibilidade para os pontos de maior probabilidade detectados pela rede neural.

5.2.4 Aplicação de filtro nos pontos detectados

Os pontos detectados pela rede neural em duas imagens com a mão na mesma posição, mas com pequenas variações, por exemplo da posição relativa da mão na região de interesse, faz com que haja pequenas variações na predição da posição pela rede neural, fazendo com que a posição em função do tempo tenha ruído. Por conta deste ruído, aplicações simples deste sistema, como a identificação da pose de garra, descrito na sessão do cliente, não eram possíveis, visto que o erro, causado pelo ruído, na posição da ponta do indicador e a do polegar eram maiores que a distância determinada como limite do estado de “Garra Fechada”.

Para melhorar a qualidade do sinal, foi-se usado o mesmo filtro utilizado em Mueller et al.(6) e definido em Casiez, Roussel e Vogel(8), chamado de *Filtro de 1 Euro*. Este filtro é utilizado por ser mais simples de implementar e definir os parâmetros que o Filtro de Kalman (30), além de ter resolvido o problema inicial de detecção de garra. Os resultados desta filtragem serão discutidos com maior profundidade no capítulo de resultados e discussões.

5.2.5 Modelo da mão

Neste ponto a aplicação tem, em relação à região de interesse, os pontos em duas dimensões das juntas. A partir destes e da posição da região de interesse em relação à imagem recebida, é possível calcular a posição dos pontos em relação à imagem inteira.

Para obter os pontos em três dimensões em relação à câmera, utilizou-se o método descrito em Panteleris, Oikonomidis e Argyros(5), o qual define este problema como um problema de cinemática inversa.

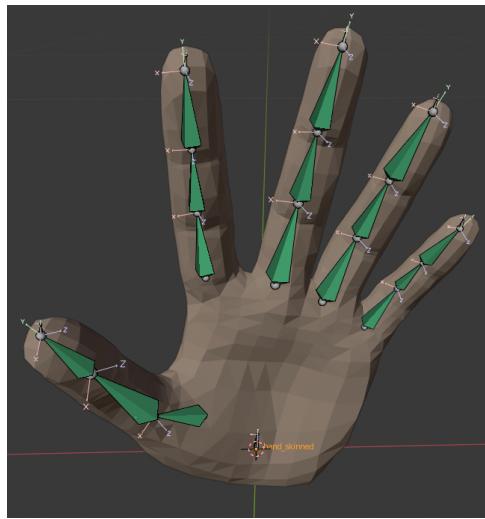


Figura 17 – Modelo da mão utilizada neste projeto com eixos de rotação visíveis. Este modelo foi obtido do projeto de Panteleris, Oikonomidis e Argyros(5).

O modelo da mão utilizado possui 26 graus de liberdade, sendo: 3 da posição do pulso; 3 da rotação do pulso, no qual cada grau de liberdade tem valores entre -180 e 180 graus; 2 para cada junta proximal, a qual pode girar entre 0 e 90 graus em torno do eixo x e entre -30 e 30 graus em torno do eixo z; 1 para cada uma das 10 juntas restantes, as quais podem girar entre 0 e 90 graus em torno do eixo x - os eixos de rotação são ilustrados na figura 17.

Assim como em Panteleris, Oikonomidis e Argyros(5), os 26 graus de liberdade são representados por 27 variáveis, uma vez que o pulso é representado por um quatérnio, em vez de 3 ângulos, a fim de evitar *Gimbal Lock*.

Para fazer a cinemática direta descrita na próxima sub-sessão, as informações que foram extraídas do modelo são:

- Os vetores unitários que representam a direção para todas as juntas ligadas, de tal forma que se forme uma arvore com o pulso como raiz e as pontas do dedo como folhas (o modulo dos vetores que ligam cada uma das juntas é calculado nos primeiros 30 quadros, considerando que a mão está estendida, como em Mueller et al.(6)).

- Os vetores unitários que representam os eixos de rotação de cada junta.

5.2.6 Cinemática Direta e Projeção dos Pontos

Para resolver o problema de cinemática inversa com o método proposto, primeiro é necessário resolver o problema de cinemática direta, ou seja, dado a posição do pulso e os ângulos de rotação quais são os pontos-chave em três dimensões e, indo além da cinemática direta, qual a projeção destes pontos na imagem.

A posição do pulso é dada na entrada. A posição das demais juntas pode ser calculada para cada junta J_i de acordo com a seguintes equações, considerando que o resultado do produto entre um quatérnio e um vetor é o vetor rotacionado nos ângulos representados pelo quatérnio, como mostrado na equação 2.6:

$$Rot_{J_i}(X) = Rot_{P_i}(X) * RotLocal_{J_i}$$

$$Pos_{J_i}(X) = Pos_{P_i}(X) + Rot_{P_i}(X) * (PosInicial_{P_i} - PosInicial_{J_i})$$

Sendo:

- P_i a junta mais próxima da raiz (pulso) na arvore que está diretamente ligada com J_i ;
- $PosInicial_{J_i}$ o vetor que representa a posição da junta i quando nenhuma rotação é aplicada às juntas.
- $RotLocal_{J_i}$ um quatérnio que representa as rotações locais da junta i;
- X uma variável de 27 dimensões que representa a posição do pulso e as rotações do pulso e das juntas, conforme descrito anteriormente;
- $Rot_{J_i}(X)$ um quatérnio que representa as rotações globais da junta i, isto é, considerando a rotação das demais juntas;
- $Pos_{J_i}(X)$ a posição final da junta i;

Com isto, obtém-se os pontos-chave em três dimensões. Para obter os pontos projetados na imagem é necessário fazer a projeção destes, usando as equações de projeção da câmera. A projeção de cada ponto p é feito de acordo com as transformações definidas em 2.10, aqui simplificadas:

$$v_1 = (-p.x/p.z, -p.y/p.z)$$

$$r = |v_1|$$

$$k = 1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6$$

$$\begin{aligned}
 v_2.x &= v_1.x * k + 2 * p_1 * v_1.x * v_1.y + p_2 * (r^2 + 2 * v_1.x^2) \\
 v_2.y &= v_1.y * k + 2 * p_2 * v_1.x * v_1.y + p_1 * (r^2 + 2 * v_1.y^2) \\
 PP &= (f_x * v_2.x, f_y * v_2.y)
 \end{aligned}$$

Sendo k_1 , k_2 e k_3 os coeficientes de distorção radial, p_1 e p_2 os componentes de distorção tangencial e PP o ponto projetado.

5.2.7 Cinemática Inversa

Considere a seguinte função $F(X)$, similar à definida em Panteleris, Oikonomidis e Argyros(5):

$$F(X) = \sum(|P_i - PP_i(X)| * c_i^3 * peso_i)^2$$

No qual:

- P_i é o ponto da junta i retornado pela rede neural;
- c_i é a confiança da previsão da junta i pela rede neural;
- $peso_i$ é um peso definido para a i -ésima junta;
- X é uma variável de 27 dimensões que representa a posição do pulso e as rotações do pulso e das juntas, conforme descrito anteriormente;
- $PP_i(X)$ é o ponto resultante da cinemática direta e projeção da junta i - este ponto é dado em função de X .

Desta forma, a pose X que gerou a mão da imagem enviada pelo cliente minimiza $F(X)$, uma vez que esta função é mínima quando $|P_i - PP_i(X)|$ é zero para cada i , ou seja, a projeção de cada junta está na mesma posição identificada pela rede neural na imagem.

Dada a forma da função, uma aproximação para o mínimo pode ser encontrada usando o método de *Levenberg-Marquardt* (19), o qual é implementado pela biblioteca utilizada neste trabalho, *ceres-solver*.

Com isto, o servidor pode retornar para o cliente a posição do pulso todas as rotações que minimizam a função e, adicionalmente, os valores dos pontos-chave das juntas em três dimensões, obtidos após realizar a cinemática direta em X .

6 RESULTADOS E DISCUSSÕES

O objetivo proposto, de realizar uma tarefa de *pick-and-place* foi realizado com sucesso, ao posicionar dois cubos coloridos nas respectivas áreas em uma cena criada em *Unity*, conforme ilustrado pela figura 14.

Neste capítulo serão apresentados os resultados obtidos neste trabalho e os dados comparativos que justificam algumas das escolhas tomadas, como a rede neural utilizada para detectar os pontos chave e o uso de um filtro.

Os gráficos das análises comparativas deste capítulo foram criados pelo mesmo conjunto de imagens de teste, ilustrados pela figura 18, gerados a partir de um vídeo de uma mão esquerda, gravada a 30 quadros por segundo, a qual está parada nos primeiros 103 quadros e, a partir destes, começa a realizar movimentos como girar e mover lateralmente, sem nenhum movimento brusco.

Estes gráficos acompanham a posição x do extremo do indicador, obtido após o passo de cinemática inversa. Este ponto está representado já no referencial da câmera, sendo x o eixo lateral no qual deslocamentos positivos são para a direita.



Figura 18 – Parte da sequência de quadros de teste, com o intervalo de 10 quadros a cada representação. Estão ordenados temporalmente da esquerda para a direita e de cima para baixo.

6.1 Análise comparativa das redes neurais

Conforme descrito nos capítulos anteriores, duas redes neurais foram utilizadas para a detecção dos pontos-chave da mão, desenvolvidas por Cao et al.(4) e Mueller et

al.(6).

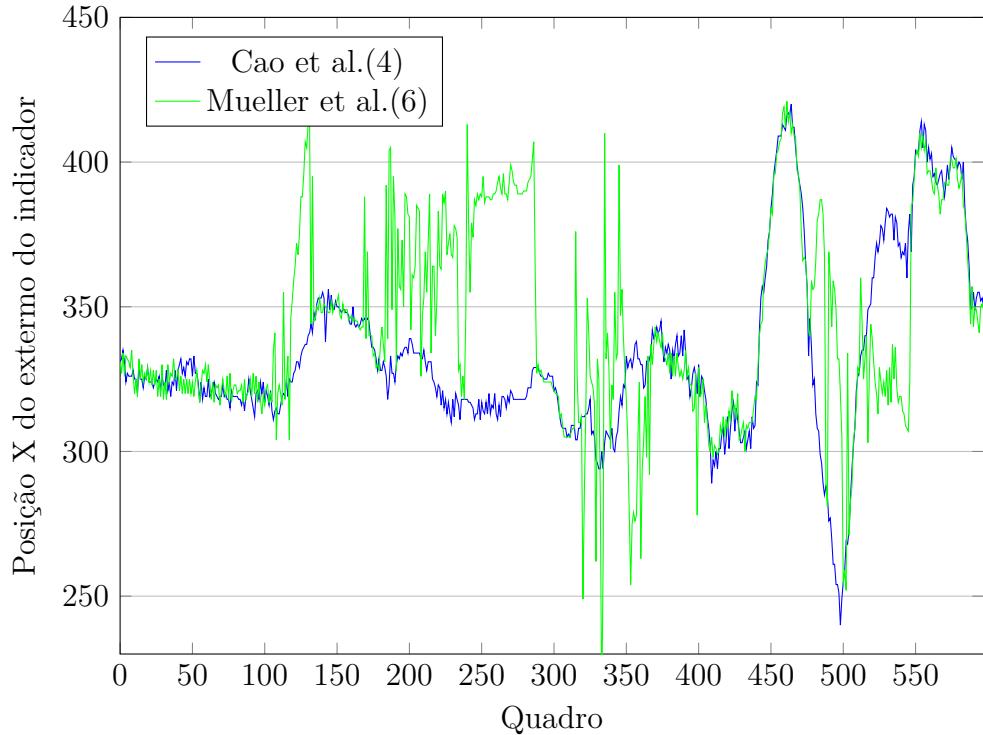


Figura 19 – Gráfico comparativo da saída do sistema da posição x do indicador, usando a rede neural de Mueller et al.(6) e a de Cao et al.(4).

Como nas imagens de teste não há movimentos bruscos, era de se esperar não haver mudanças grandes na posição do extremo do indicador, ao contrário do resultado adquirido usando a rede de Mueller et al.(6). Já a rede de Cao et al.(4), teve um resultado mais próximo do esperado. Além disto, nos testes qualitativos realizados, a pose do esqueleto tridimensional obtido por esta rede é mais próxima do esperado, enquanto a de Mueller et al.(6) apresenta grandes variações, por exemplo trocando frequentemente a posição do extremo do indicador com o extremo do dedo médio.

Apesar da rede de Mueller et al.(6) ser mais imprecisa, esta é muito mais rápida, levando em média 7.77 ms para ser executada enquanto a rede de Cao et al.(4) leva em torno de 72.71 ms. Mesmo com esta diferença, a qual impacta significativamente a quantidade de quadros por segundo que podem ser processados, preferiu-se usar a segunda rede, uma vez que a tarefa proposta de *pick-and-place* não era possível com a primeira. É possível que se obtenha resultados superiores aos de Cao et al.(4), pelo menos para a tarefa citada, se a rede de Mueller et al.(6) for treinada com poses que refletem esta tarefa. Outra alternativa é utilizar outras redes como a de Zhang et al.(31).

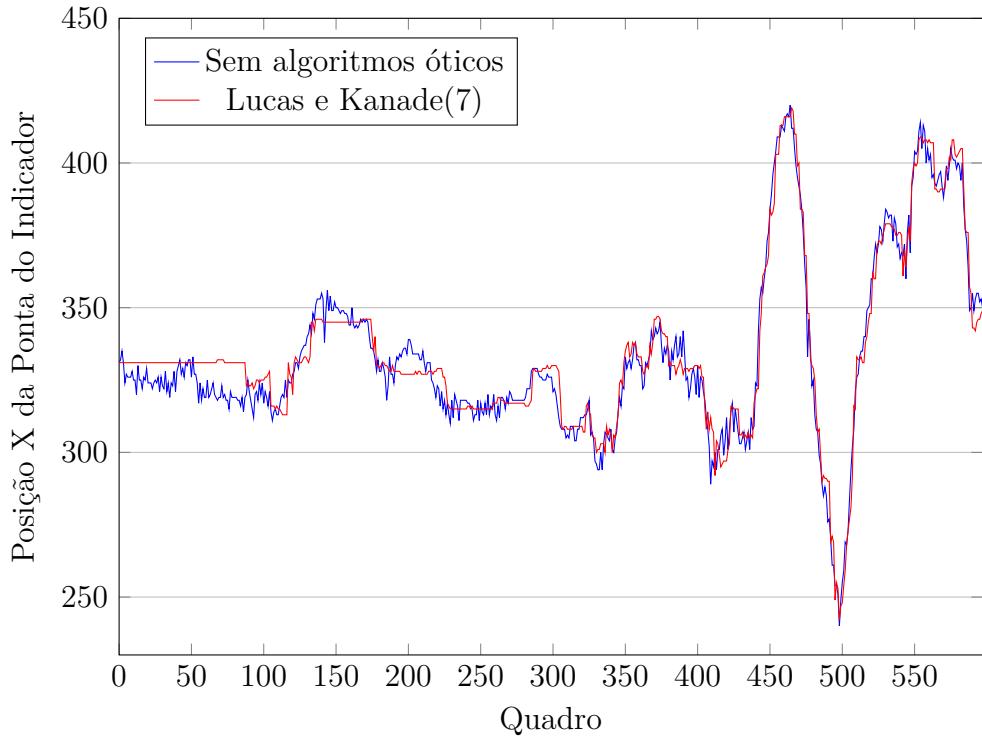


Figura 20 – Gráfico comparativo da saída do sistema da posição x do indicador, com e sem o uso do algoritmo ótico de Lucas e Kanade(7).

6.2 Análise do desempenho de algoritmos para diminuir ruído

Inicialmente, mesmo optando pelo uso da rede neural descrita na sessão anterior com maior precisão, a tarefa de *pick-and-place* era difícil de ser executada. O motivo principal disso era que o esqueleto como um todo tremia com alta frequência e, consequentemente, a distância entre o extremo do indicador e do polegar variava muito. Esta variação refletia a mudança do ponto de maior probabilidade de cada um dos pontos chaves.

Para diminuir o tremor, foi testado usar um filtro passa baixa de Casiez, Roussel e Vogel(8), e usar o método de Lucas e Kanade(7), de fluxo ótico.

O gráfico da figura 6.2 foi gerado usando a implementação do algoritmo da biblioteca *OpenCV*, com o tamanho de janela igual a 50x50, uma pirâmide de 3 níveis e a flag *OPTFLOW_USE_INITIAL_FLOW*. Apesar do gráfico mostrar que há um aumento na estabilidade dos pontos-chave, em especial da ponta do indicador, pode-se notar que isso resulta em uma estabilidade maior que a desejada. Isto pode ser notado nos primeiros 70 quadros, em que a posição tem um valor fixo e, mesmo ignorando o ruído de alta frequência, pode-se notar que o valor da posição deveria lentamente diminuir.

Fazendo um ajuste mais fino dos parâmetros deste algoritmo, é possível obter resultados melhores para a posição do extremo do indicador. No entanto, este método depende de pontos de alto contraste e, especialmente para pontos como o pulso, isto não

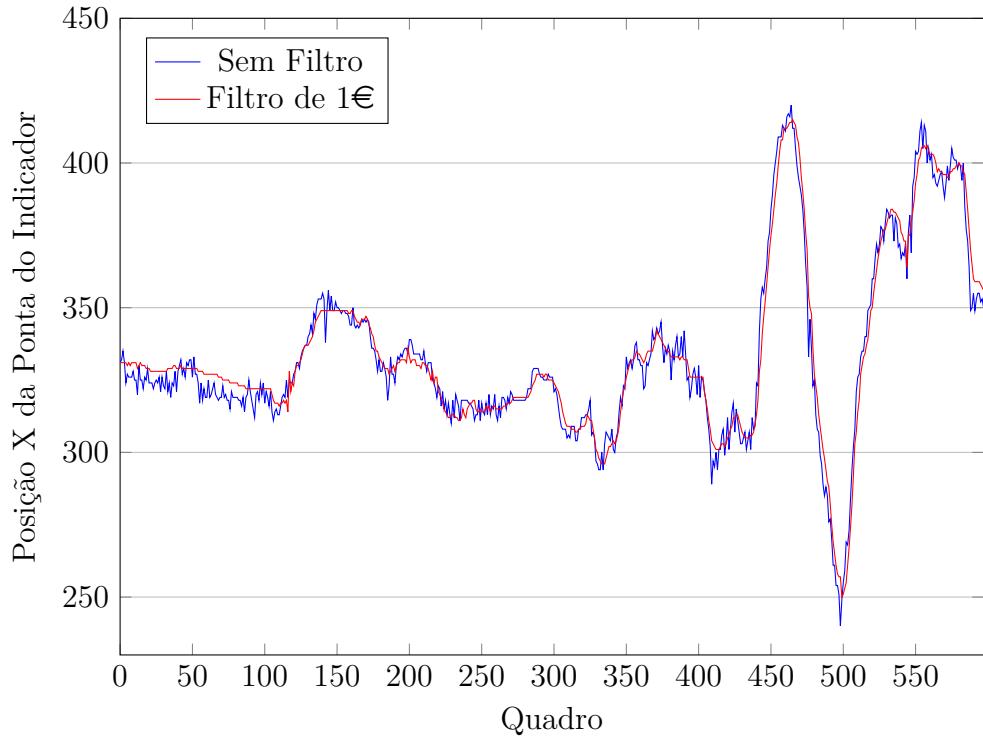


Figura 21 – Gráfico comparativo da saída do sistema da posição x do indicador, com e sem o uso do filtro de Casiez, Roussel e Vogel(8).

pode ser garantido.

O segundo método testado para diminuir o ruído foi o filtro desenvolvido por Casiez, Roussel e Vogel(8), o qual foi utilizado no trabalho de Mueller et al.(6). O resultado do filtro implementado, tomando como base uma das implementações referenciadas por Casiez, Roussel e Vogel(8), pode ser observado no gráfico da figura 6.2. Nesta implementação foi-se usada a frequência mínima igual a 10, o corte mínimo igual a 0.5 e o valor de beta igual a 0.007.

A aplicação do filtro com estes parâmetros apresentou resultados satisfatórios - suficientes para possibilitar a tarefa de *pick-and-place* e não haver mais tremores aparentes - e, como pode ser observado pelo gráfico, não ocorreu o problema de estabilização excessiva. Como os resultados foram suficientes, não foram testados outros filtros mais complexos e pesados computacionalmente como o filtro de Kalman(30).

6.3 Análise do tempo de execução do sistema integrado

Os testes realizados no desenvolvimento desse trabalho alcançaram localmente 5.37 quadros por segundo e no celular 4.20 quadros por segundo. Apesar da pose da mão não atualizar sempre. Desta forma, a imersão não é severamente afetada.

Dada a importância da taxa de atualização para imersão, são apresentados os

tempos de execução de diversas partes, desde a captura do quadro até a renderização da pose. Os dados apresentados foram obtidos usando como cliente um celular One Plus 7, o qual possui um *Chipset Snapdragon 855 Qualcomm SDM855*, um processador *1x 2.84 GHz Kryo 485 + 3x 2.42 GHz Kryo 485 + 4x 1.8 GHz Kryo 485* e uma *GPU Adreno 640*. E um notebook *HP OMEN - 15-dc0030np* com um processador *Intel Core i7-8750H* e uma placa de vídeo *NVIDIA GeForce GTX 1050 Ti*.

Tarefa	Tempo Médio de Execução (ms)
Codificação (Cliente)	12.98
Decodificação (Servidor)	1.50
Captura de Quadro (Cliente)	4.37
Operações no Quadro (Cliente)	55.70
Outros Processamentos (Cliente)	0.51

Tabela 1 – Tempo médio de execução de tarefas realizadas no cliente e no servidor. Estes dados foram obtidos fazendo testes com duração de 200 quadros.

A tabela 1 mostra que grande parte do tempo gasto no cliente está relacionado com operações realizadas no quadro, como inverter a imagem e remover o canal de alfa. Esta tarefa é realizada usando apenas o processador em uma única *thread*, é possível que este tempo diminua caso se use a placa de vídeo para realizar estes cálculos com alta paralelização.

Outro fato notável é o tempo usado para fazer a codificação e decodificação com o algoritmo *VP8* que é somado ao tempo total mas, conforme é notado na tabela 2, este tempo é compensado durante a transferência em conexões não locais, principalmente no caso do uso da conexão *Wi-Fi*.

Na tabela 2 também pode ser notado que o tempo de transferência é responsável por no mínimo 68.62 ms, em testes não locais, o qual por si só faz com que a quantidade de quadros por segundo seja no máximo 14.57. Ainda nesta tabela, pode-se notar que o tempo de processamento do servidor diminui em testes não locais, possivelmente por não fazer os processamentos do cliente também.

6.4 Análise do tempo de execução do Servidor

A tabela 3 mostra o tempo de cada uma das principais tarefas do servidor para cada quadro recebido. Estes tempos de execução foram medidos usando um modo de teste que lê diretamente os quadros citados no inicio do capítulo sem se conectar com um cliente.

A primeira tarefa, a detecção da região de interesse, tem um tempo médio de execução relativamente baixo e pode ser justificado por não usar a rede de Redmon e Farhadi(1) em todos os quadros, apenas quando a confiança na predição da posição dos pontos-chave é baixa.

Conexão	Tempo Médio de Execução (ms)		
	Cliente	Servidor	Transferência
Local	196.29	109.68	28.52
Local - Sem <i>VP8</i>	187.34	108.03	41.54
USB 3.0	241.75	92.97	68.62
USB 3.0 - Sem <i>VP8</i>	269.06	97.41	110.52
WiFi	238.35	91.73	80.29
WiFi - Sem <i>VP8</i>	316.42	97.03	158.74

Tabela 2 – Tempo de execução médio para cada quadro, por tipo de conexão com e sem o uso de *VP8*, algoritmo de compressão de mídia em tempo real. Os dados locais foram obtidos usando os mesmos quadros de teste citados no inicio deste capítulo e os demais foram gerados a partir de testes com duração de 200 quadros.

Tarefa	Tempo Médio de Execução (ms)
Detecção da Região de Interesse	2.05
Detecção das Juntas - Cao et al.(4)	72.71
Detecção das Juntas - Mueller et al.(6)	7.77
Filtro de Casiez, Roussel e Vogel(8)	<0.01
Algoritmo de Lucas e Kanade(7)	1.69
Cinemática Inversa	10.95

Tabela 3 – Tempo médio de execução das tarefas do servidor durante o processamento de cada quadro.

A tabela 3 também evidencia o baixo custo computacional do uso do filtro de 1 Euro, visto que ele é executado em apenas 21 pontos, enquanto o algoritmo de Lucas e Kanade(7) tem um custo centenas de vezes maior, visto que processa duas imagens para obter o resultado.

O passo de cinemática inversa é o mais custoso do lado do servidor, depois apenas da detecção das juntas na imagem, e leva 10.95 ms para ser executado. Outras durações, como o de processamento da imagem no cliente, são maiores e aparentam ter maiores possibilidades de melhoria.

7 CONCLUSÃO

O fator de imersão é uma das principais preocupações no desenvolvimento de uma boa experiência VR. O problema de *Hand-Tracking* em três dimensões por si só é um grande desafio tecnológico contemporâneo. Integrar essa tecnologia em ambientes de VR possibilita a criação de aplicações que melhoram a imersão nestes ambientes. Com a decisão de ter uma solução de baixo custo e não incluir sensores nas mãos do usuário, o uso de aparelhos mais caros e completos como câmeras RGB-D ou infravermelho foi intencionalmente descartado, o que tornou o desafio ainda maior. Tendo como restrição usar apenas uma câmera RGB, surgiu a necessidade de utilizar técnicas recentemente criadas que estão no estado da arte.

Como resultado deste trabalho, foi possível criar uma solução de baixo custo que captura quadros pela câmera de um celular inserido em um *Google Cardboard*, o qual é processado em um servidor e enviado de volta para a aplicação VR. Dessa forma, o usuário pode visualizar a própria mão em um ambiente virtual e interagir com o mesmo. A capacidade de interação foi confirmada ao realizar o objetivo proposto na introdução - realizar uma tarefa de *pick-and-place*, movendo um cubo para uma área específica.

Os equipamentos adicionais necessários, além de um celular e de um computador, são a estrutura de papelão e as lentes do *Google Cardboard*, implicando em um baixo custo para usar o sistema criado.

Este trabalho usa o *Google Cardboard* como base, mas a modularidade deste permite que outras aplicações de realidade virtual, ou mesmo de realidade aumentada, possam ser integradas. Dessa forma, os desenvolvedores VR que utilizarem esta ferramenta poderão abstrair o conhecimento necessário de visão computacional para *hand-tracking* em três dimensões - é necessário apenas fornecer quadros para receber a pose da mão. Além disso, por ser *open-source*, outros desenvolvedores podem contribuir de forma direta implementando novas funcionalidades ou mesmo utilizando e comentando pontos a serem melhorados.

Algumas oportunidades de melhoria podem ser exploradas em trabalhos futuros, principalmente as que tangem o tempo de processamento, uma vez que é um fator importante para o aumento da imersão no ambiente virtual.

A maior parte das tarefas, desde a captura do quadro até a atualização da pose no cliente, são executadas para apenas um quadro de cada vez, isto é, um novo quadro só começa a ser processado quando a pose no cliente é atualizada. Uma melhora, baseando-se nisto, é paralelizar mais as tarefas, transformando em uma *pipeline*, tanto do lado do servidor quanto do cliente.

Outra possibilidade para diminuir o tempo de processamento está associado ao teste usando redes neurais mais leves como a desenvolvida por Zhang et al.(31), ou mesmo mudar o conjunto de testes da rede de Mueller et al.(6) para refletir melhor a tarefa de *pick-and-place* e ter um desempenho melhor.

Conforme mencionado no capítulo de Resultados e Discussões, o tempo gasto no cliente com operações nos quadros também tem a possibilidade de ser diminuído ao usar a placa de vídeo no lugar do processador, uma vez que a codificação, por exemplo, também tem que realizar operações no quadro e possui um tempo de processamento quase cinco vezes menor. Além disto, há a possibilidade de diminuir o tempo total de transferência usando protocolos de transferência mais simples de camadas mais baixas, em vez do protocolo HTTP, e comprimir os dados usando algum algoritmo otimizado para o *hardware* do cliente e do servidor.

A tarefa de *pick-and-place* proposta neste trabalho também pode ser explorada em outros cenários ou mesmo pode-se generalizar os estados de garra para uma máquina de estados que permita movimentos mais complexos para interagir com o ambiente virtual.

REFERÊNCIAS

- 1 REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- 2 KUCUK, S.; BINGUL, Z. *Robot kinematics: Forward and inverse kinematics*. London, UK: INTECH Open Access Publisher, 2006.
- 3 STRICKLAND, J. What is a gimbal—and what does it have to do with nasa. *HowStuffWorks. com*, v. 20, 2008.
- 4 CAO, Z.; SIMON, T.; WEI, S.-E.; SHEIKH, Y. Realtime multi-person 2d pose estimation using part affinity fields. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. San Franscisco, CA: IEEE, 2017. p. 7291–7299.
- 5 PANTELERIS, P.; OIKONOMIDIS, I.; ARGYROS, A. Using a single rgb frame for real time 3d hand pose estimation in the wild. In: IEEE. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Lake Tahoe, NV/CA, 2018. p. 436–445.
- 6 MUELLER, F.; BERNARD, F.; SOTNYCHENKO, O.; MEHTA, D.; SRIDHAR, S.; CASAS, D.; THEOBALT, C. Ganerated hands for real-time 3d hand tracking from monocular rgb. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. San Francisco, CA: IEEE, 2018. p. 49–59.
- 7 LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981. (IJCAI'81), p. 674–679.
- 8 CASIEZ, G.; ROUSSEL, N.; VOGEL, D. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY: ACM, 2012. p. 2527–2530.
- 9 BURDEA, G. C.; COIFFET, P. *Virtual reality technology*. [S.l.]: John Wiley & Sons, 2003.
- 10 SHERMAN, W. R.; CRAIG, A. B. *Understanding virtual reality: Interface, application, and design*. Cambridge, MA: Morgan Kaufmann, 2018. 582 p.
- 11 BARON, R. J. Mechanisms of human facial recognition. *International Journal of Man-Machine Studies*, Elsevier, v. 15, n. 2, p. 137–178, 1981.
- 12 FUJIYOSHI, H.; HIRAKAWA, T.; YAMASHITA, T. Deep learning-based image recognition for autonomous driving. *IATSS research*, Elsevier, v. 43, n. 4, p. 244–252, 2019.
- 13 MOHANTY, S. P.; HUGHES, D. P.; SALATHÉ, M. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, frontiers, v. 7, p. 1419, 2016.
- 14 KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105.

- 15 HAMILTON, W. R. *Elements of quaternions*. [S.l.]: Longmans, Green, & Company, 1866.
- 16 FINKELSTEIN, D.; JAUCH, J. M.; SCHIMINOVICH, S.; SPEISER, D. Foundations of quaternion quantum mechanics. *Journal of mathematical physics*, American Institute of Physics, v. 3, n. 2, p. 207–220, 1962.
- 17 SCHMIDT, J.; NIEMANN, H. Using quaternions for parametrizing 3-d rotations in unconstrained nonlinear optimization. In: CITESEER. *Vmv*. [S.l.], 2001. v. 1, p. 399–406.
- 18 Sanderson, G.; Eater, B. *Visualizing Quaternions, An Explorable Video Series*. 2020. Disponível em: <<https://eater.net/quaternions/>>.
- 19 MORÉ, J. J. The levenberg-marquardt algorithm: implementation and theory. In: *Numerical analysis*. Berlin, Heidelberg, DE: Springer, 1978. p. 105–116.
- 20 CALIBRATION, C. 3d reconstruction, opencv 2.4. 13.6 documentation. *Opencv dev team*, 2018.
- 21 IWASAWA, S.; EBIHARA, K.; OHYA, J.; MORISHIMA, S. Real-time estimation of human body posture from monocular thermal images. In: IEEE. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.], 1997. p. 15–20.
- 22 EISEN-ENOSH, A.; FARAH, N.; BURGANSKY-ELIASH, Z.; POLAT, U.; MANDEL, Y. Evaluation of critical flicker-fusion frequency measurement methods for the investigation of visual temporal resolution. *Scientific reports*, Nature Publishing Group, v. 7, n. 1, p. 15621, 2017.
- 23 TYLER, C. W. Analysis of visual modulation sensitivity. iii. meridional variations in peripheral flicker sensitivity. *JOSA A*, Optical Society of America, v. 4, n. 8, p. 1612–1619, 1987.
- 24 ZIMMERMANN, C.; BROX, T. Learning to estimate 3d hand pose from single rgb images. In: *The IEEE International Conference on Computer Vision (ICCV)*. [s.n.], 2017. Acesso em: 21/05/2019. Disponível em: <http://openaccess.thecvf.com/content_iccv_2017/html/Zimmermann_Learning_to_Estimate_ICCV_2017_paper.html>.
- 25 ZHANG, J.; JIAO, J.; CHEN, M.; QU, L.; XU, X.; YANG, Q. 3d hand pose tracking and estimation using stereo matching. *arXiv preprint arXiv:1610.07214*, v. 1, 2016. Acesso em: 21/05/2019. Disponível em: <<https://arxiv.org/abs/1610.07214>>.
- 26 GOLDSTONE, W. *Unity game development essentials*. Birmingham, UK: Packt Publishing Ltd, 2009.
- 27 MISHRA, P.; SHRAWANKAR, U. Comparison between famous game engines and eminent games. *International Journal of Interactive Multimedia & Artificial Intelligence*, v. 4, n. 1, 2016.
- 28 Sharrab, Y. O.; Sarhan, N. J. Detailed comparative analysis of vp8 and h.264. In: IEEE. *2012 IEEE International Symposium on Multimedia*. Irvine, CA, 2012. p. 133–140.

- 29 Chen, Y.; Murherjee, D.; Han, J.; Grange, A.; Xu, Y.; Liu, Z.; Parker, S.; Chen, C.; Su, H.; Joshi, U.; Chiang, C.; Wang, Y.; Wilkins, P.; Bankoski, J.; Trudeau, L.; Egge, N.; Valin, J.; Davies, T.; Midtskogen, S.; Norkin, A.; de Rivaz, P. An overview of core coding tools in the av1 video codec. In: IEEE. *2018 Picture Coding Symposium (PCS)*. San Francisco, CA: arXiv, 2018. p. 41–45.
- 30 KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, v. 82, n. 1, p. 35–45, 03 1960.
- 31 ZHANG, F.; BAZAREVSKY, V.; VAKUNOV, A.; TKACHENKA, A.; SUNG, G.; CHANG, C.-L.; GRUNDMANN, M. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.