



**POSICIONAMENTO E EXIBIÇÃO DE IMAGENS 3D UTILIZANDO
ÓCULOS DE REALIDADE AUMENTADA PARA APLICAÇÃO
CIRÚRGICA**

Relatório final na modalidade de auxílio à iniciação científica, submetido à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Processo: 2020/15835-4

Pesquisador Responsável: Dr. Glauco Augusto de Paula Caurin

Coorientador: Paulo Henrique Polegato

Beneficiário: Calvin Suzuki de Camargo

Informações gerais do projeto

- Título do projeto:

Posicionamento e exibição de imagens 3D utilizando óculos de realidade aumentada para aplicação cirúrgica

- Pesquisador responsável:

Dr. Glauco Augusto de Paula Caurin

- Coorientador:

Paulo Henrique Polegato

- Beneficiário:

Calvin Suzuki de Camargo

- Número do processo do projeto:

2020/15835-4

- Instituição sede do projeto:

Escola de Engenharia de São Carlos da Universidade de São Paulo

- Período de vigência proposto:

01 de março de 2021 a 28 de fevereiro de 2022

- Período coberto por este relatório científico:

01 de março de 2021 a 28 de fevereiro de 2022

Resumo

Trata-se do relatório final do projeto de pesquisa de iniciação científica que comprehende os trabalhos iniciados em março de 2021 até o final de fevereiro de 2022. O projeto visa estudar a exibição de objetos 3D em óculos inteligentes (*Smart glasses*) com realidade aumentada, que podem servir como um dispositivo auxiliar para aplicações cirúrgicas. Para isso, pretende-se conceituar as relações entre visão computacional e computação gráfica no campo da realidade aumentada. Com o apoio do Laboratório Aeronáutico de Tecnologias (AeroTech), estudaremos uma forma de elaborar uma aplicação que sobreponha a cabeça de um paciente com um modelo 3D indicando pontos de implantação de eletrodos durante o procedimento neurocirúrgico orientado por estereoeletroencefalografia (SEEG).

Palavras-chaves: *Smart glasses*, Aplicação cirúrgica, Visão computacional, Realidade aumentada.

Sumário

Informações gerais do projeto	i
Resumo	ii
1 Resumo do projeto	1
1.1 Enunciado do problema	1
1.2 Objetivo	2
1.3 Metodologia	2
1.4 Breve histórico do projeto	3
2 Realizações	4
2.1 Moverio BT-350	4
2.1.1 Documentação	4
2.1.2 <i>Calibration Tool</i>	5
2.1.3 <i>Capture Tool</i>	5
2.1.4 <i>Training Tool</i>	5
2.1.5 <i>Moverio AR</i>	7
2.2 Estudos em desenvolvimento Android Studio	7
2.3 Estudos de desenvolvimento Unity	9
2.4 Revisão bibliográfica	11
2.5 Elaboração do VCraniun	12

2.5.1	Arquitetura do Sistema	12
2.5.2	Comunicação	14
2.5.3	Estimação da posição da projeção	15
2.5.4	<i>User Interface</i>	16
Referências bibliográficas		17

1 Resumo do projeto

1.1 Enunciado do problema

As cirurgias na medicina têm também o objetivo de causar a mínima agressão ao organismo do paciente, isso implica em um menor tempo de hospitalização; menor incidência de complicações pós-operatórias; menos dor; e recuperação mais rápida. Contudo, entre os diversos desafios de sua realização, a habilidade visual do médico, que envolve a capacidade de localizar e identificar os tecidos sensíveis, é essencial para redução dos danos ao paciente durante um procedimento cirúrgico, nos levando à questão: Como podemos auxiliar a visão do médico para aumentar a qualidade dessas operações? Para esse fim, um *software* que informa e assiste o médico em tempo real durante a operação denota-se como um ótimo meio de aumentar as chances de sucesso dessas cirurgias.

Atualmente, *softwares* auxiliares já são muito presentes no cotidiano de trabalho dos cirurgiões, entretanto, toma-se como exemplo os sistemas de neuronavegação, cuja função é exibir com precisão as estruturas do sistema nervoso do paciente. Entretanto, o médico, um profissional que foi treinado para ler esses dados, ainda precisa realizar um esforço na leitura da neuronavegação durante a cirurgia, colaborando para um maior tempo de operação e favorecendo a exaustão



Figura 1.1: Exemplo da utilização de neuronavegador em cirurgia. O cirurgião precisa tirar os olhos do paciente e interpretar as imagens do monitor, impactando negativamente a performance do médico. Fonte: (SERVIÇO..., 2019).

do profissional, de modo que aumente seu potencial de erros durante os procedimentos cirúrgicos (1.1). Para isso, a tecnologia de realidade aumentada (*AR*) oferece o potencial de reduzir essas limitações, em razão de que figuras tridimensionais podem sobrepor a visão do médico para facilitar a visualização e a localização dos objetivos da operação enquanto tem seus olhos direcionados ao paciente (CHO et al., 2020).

O que se procura para este projeto é uma pesquisa que envolve visão computacional e computação gráfica no campo da realidade aumentada. Para isso, é importante escolher um sistema de baixa complexidade que colabore para a aplicação direta dos conceitos estudados, portanto, a escolha dos óculos da empresa *Seiko Epson Corporation*[®], modelo *Moverio BT-350™*, correlaciona-se com esse objetivo, pois é uma opção que sucintamente leva o *AR* para o cirurgião, envolve o mínimo de aparelhagem e, ainda assim, preserva a robustez no sistema. Recebendo suporte físico e técnico da equipe do Laboratório Aeronáutico de Tecnologias (Aerotech) da Universidade de São Paulo, esse estudo corrobora com o objetivo do grupo que é desenvolver uma plataforma colaborativa de multi-aplicação cirúrgica.

1.2 Objetivo

Pretende-se exibir informações e posicionar modelos tridimensionais em uma região do espaço com *AR*, de forma que facilite o acesso do cirurgião à informação durante a cirurgia; estudar e registrar a resposta dos equipamentos utilizados no quesito de qualidade gráfica e latência de resposta do sistema. Tudo isso, com o objetivo central de aumentar a proximidade do cirurgião com a tecnologia de *AR* como apoio durante os procedimentos cirúrgicos.

1.3 Metodologia

Consiste na listagem de possíveis soluções, técnicas ou ferramentas; o estudo e a discussão sobre elas, em seguida, sua implementação. Paralelamente a isso, a busca bibliográfica é constantemente realizada com o objetivo de esclarecer dúvidas sobre os meios imaginados e discutidos com o orientador e coorientador. Essa busca dá ênfase nos resultados encontrados pelos artigos, o objetivo disto é caracterizar os prós e contras das diversas opções encontradas na

listagem de técnicas e soluções. Essa pesquisa de artigos trás uma evolução do discernimento dos assuntos do estado da arte, refletindo em uma noção de funcionamento dos métodos científicos que constrói um novo pesquisador no desenvolvimento dessa iniciação científica.

1.4 Breve histórico do projeto

Desde o início dos trabalhos no projeto, foram experimentados diversos tipos de contato com a elaboração de *softwares* para o sistema operacional *Android* (1.2a); testes das ferramentas da documentação dos óculos de realidade aumentada *SEIKO EPSON Corporation™ Moverio BT-350* (1.2b e 1.2c); e a elaboração de aplicativos que ilustram o objetivo do *VCranium* (1.2d). Assim, como foi explicado no relatório parcial da pesquisa, pretendíamos prosseguir o desenvolvimento estabelecendo uma arquitetura composta por computador, câmera (*webcam*) e óculos para capturar os dados necessários para a projeção em realidade aumentada, os detalhes serão descritos no capítulo das realizações.



Figura 1.2: Histórico de realizações da pesquisa até a primeira entrega parcial. Fonte: Autor.

2 Realizações

2.1 Moverio BT-350

2.1.1 Documentação

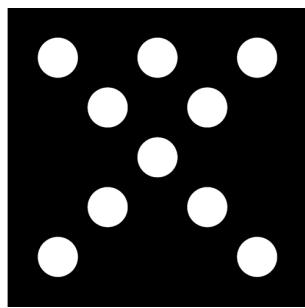
A *Epson* disponibilizou uma documentação sobre as funcionalidades dos óculos e forneceu algumas ferramentas e aplicações exemplares para auxiliar os desenvolvedores de aplicativos do *Moverio*. A tabela abaixo apresenta os arquivos da documentação e uma breve descrição:

Arquivo	Descrição
<i>CalibrationTool.apk</i>	Aplicativo de calibração da projeção da tela dos óculos
<i>CaptureTool.apk</i>	Ferramenta que ajuda a capturar fotos para o mapeamento de novos marcadores e objetos
<i>TrainingToolWindows</i>	Usa as imagens obtidas do <i>CaptureTool.apk</i> e cria o <i>dataset</i> para a detecção do novo marcador ou objeto
<i>Moverio_AR</i>	Exemplo de cena que aplica funções do <i>Moverio</i> no <i>Unity</i>

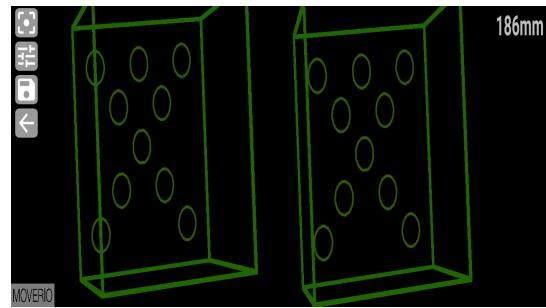
Antes de prosseguirmos com os testes de cada programa, é importante denotar que o suporte da fabricante foi precário. No momento em que o projeto foi aprovado e acessou os *sites* da fabricante, não foi encontrado a atualização de sistema, ficando com o *Android 5.1 (API 22 Lollipop)* que estava muito desatualizado em relação ao mercado atual. Além disso, existia uma biblioteca exclusiva de aplicativos para o *Moverio*, porém não para o *BT-350*, mas para outros modelos com foco em usuário. A documentação existia, porém sua última atualização foi em julho de 2019, segundo os metadados dos arquivos. Por fim, infelizmente, o suporte do produto foi descontinuado enquanto a pesquisa utilizava sua documentação, ademais, os arquivos da documentação foram excluídos do *site* da fabricante. O *download* de todos os arquivos foi feito e reservado para os estudantes do laboratório.

2.1.2 *Calibration Tool*

Segundo a fabricante, cada pessoa tem uma distância e um formato de rosto diferente, exigindo uma calibração específica para cada usuário (MARUYAMA et al., 2018). Essa calibração pode ser feita manualmente por meio da interface do *CalibrationTool.apk*, que é instalado diretamente nos óculos. A calibração é efetuada ao ajustarmos uma imagem com o movimento da cabeça sobre o marcador da Figura 2.1a. O processo é feito com cada olho, então fechamos o olho direito, ajustamos o esquerdo e fazemos o mesmo para o direito (Figura 2.1b).



(a) Marcador da biblioteca *Moverio*



(b) Aplicativo fazendo a calibração

Figura 2.1: Calibração usando marcador impresso e o aplicativo *Calibration Tool* instalado nos óculos. Fonte: Autor.

2.1.3 *Capture Tool*

A *Epson* forneceu um aplicativo para auxiliar a captura de fotos para o reconhecimento de novos objetos. O programa utiliza a imagem do marcador (Figura 2.1a) que serve de referência de localização para a câmera que captura 16 imagens com a posição pré-definida. A obturação é automática quando o usuário entra na posição determinada, no entanto, a utilização do programa não é satisfatória na maioria das vezes: A câmera aciona automaticamente quando você está indo para a posição desejada, ou seja, ele obtura uma imagem embaçada pois sua cabeça ainda está em movimento.

2.1.4 *Training Tool*

O *software* tem o intuito de ampliar as possibilidades da detecção adicionando novos marcadores e objetos nos programas elaborados. Para treinarmos um objeto precisamos importar

o modelo 3D (*STL*) no programa, e então utilizar as imagens coletadas do *Capture Tool* para fazer um treinamento que habilitará a identificação do novo objeto. O primeiro teste foi realizado com uma lata de refrigerante, encontrado gratuitamente na *internet*. Quando prosseguimos no programa, é pedido para escolhermos uma das imagens da captura e correlacionar pontos no objeto real da imagem e o modelo virtual (Figura 2.2). Após esse processo manual, o programa realiza a sobreposição do modelo no resto das imagens e libera uma ação de ajuste finais.

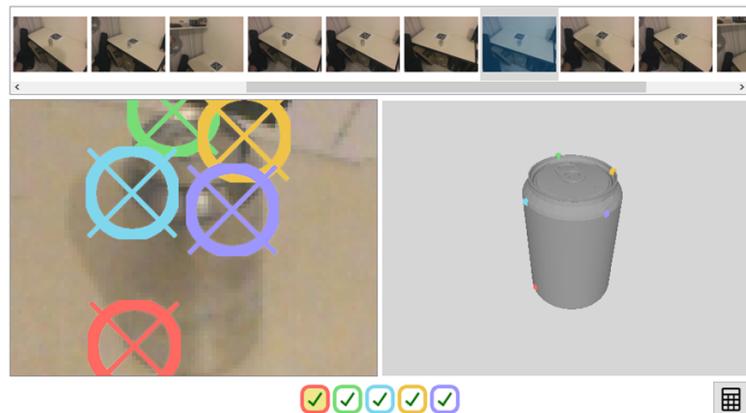


Figura 2.2: Tela da seleção de pontos. Fonte: Autor.

O ajuste manual funciona muito bem, seu resultado é a figura 2.3a. O próximo passo é esperar o programa encontrar a posição automaticamente no resto das imagens, porém, isso não ocorre como o desejado. A Figura 2.3b representa uma falha na estimativa de posição da lata, normalmente, pensaríamos em descartar essa imagem da lista, no entanto, essa é a melhor tentativa que o programa fez, os demais erram tanto que a latinha mal aparece na imagem. Ao tentar realizar um reajuste nessa imagem, o programa impede a ação e notifica que a diferença é muito grande em relação ao primeiro ajuste, o que impossibilita o resultado de ser satisfatório.



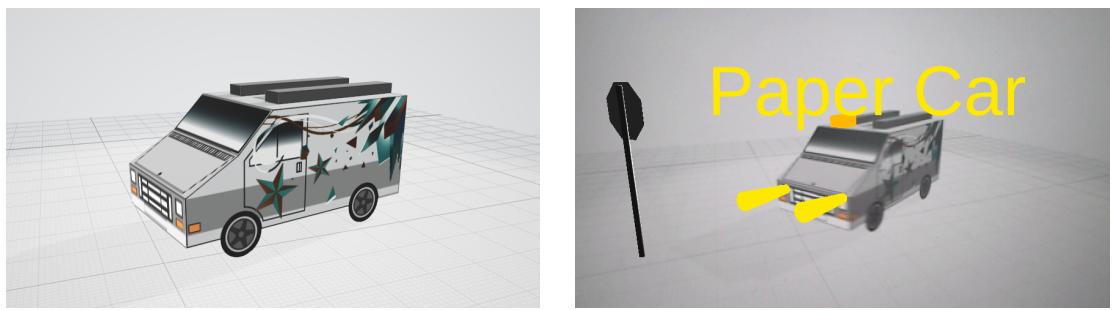
(a) Ajuste manual da Figura 2.2

(b) Posição encontrada pelo programa

Figura 2.3: Tentativa do treinamento na detecção da lata de refrigerante. Fonte: Autor.

2.1.5 Moverio AR

Unity é um programa voltado para o desenvolvimento de jogos em múltiplas plataformas, sendo o *Android* uma delas, e com o apoio da USP, temos acesso a mais recursos para os projetos desenvolvidos nele (UNITY..., 2021). A documentação da forneceu um programa de exemplo para *Unity* que pode reconhecer um carrinho de papel e sobrepor com animações em AR. Para o teste, supomos que ele funcionaria com uma imagem no computador, pois ele não pode diferenciar tridimensionais e bidimensionais pela limitação da câmera. De fato, isso ocorreu e o teste foi registrado na figura 2.4.



(a) Modelo virtual do carrinho de papel (b) Sobreposição em AR com *Moverio*

Figura 2.4: Resultados adquiridos dos testes da documentação *Moverio* no *Unity*. Fonte: Autor.

2.2 Estudos em desenvolvimento Android Studio

O desenvolvimento de aplicativos para *Android* foi estudado nos primeiros meses da pesquisa. Nessa parte do projeto, a pesquisa teve um aspecto mais técnico, que foi necessário para a programação das futuras aplicações que estariam por vir. A *IDE* (ambiente de desenvolvimento integrado) utilizada foi o *Android Studio*, que foi sugerida pelo curso adquirido da *Udemy* (UDEMY..., 2021).

Após as primeiras semanas de aprendizado *Android*, foi iniciado uma pesquisa sobre o desenvolvimento de aplicativos voltados a realidade aumentada. Porém, antes de irmos diretamente nisso, um interessante teste foi realizado com *OpenCV*, que consistiu em realizar a primeira manipulação das imagens da câmera (ABOUT..., s.d.). A experiência foi proveitosa para o aprendizado da integração de ferramentas externas ao projeto padrão da plataforma e que, ademais, será feito muitas vezes até a conclusão da pesquisa.

A primeira biblioteca de realidade aumentada a ser testada foi o *Google Sceneform* (SCE-NEFORM..., 2021). Muitos problemas foram encontrados na integração dos *plugins* com o *Android Studio*, pois este precisava estar em uma versão antiga específica para funcionar. Após a instalação, foi possível ver a primeira projeção em realidade aumentada em um simulador de *smartphone* no computador (figura 2.5a). Restava testar o aplicativo para um *smartphone* real, porém o afastamento dos integrantes do laboratório pela pandemia, e a falta de dispositivos compatíveis à minha disposição, causaram um atraso nos testes. Felizmente, foi gerado um arquivo que permite a instalação à distância e o aplicativo funcionou com sucesso nos celulares da equipe (figura 2.5b).

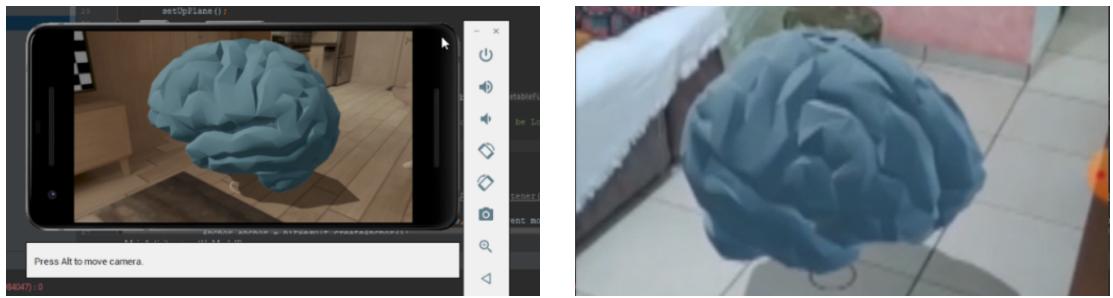


Figura 2.5: Resultados adquiridos com o *Google Sceneform*. Fonte: Autor.

Prosseguindo os estudos de aplicativos AR, percebemos que os problemas de versão tidos com o *Sceneform* estavam sendo reparados pelo *Google* e adaptados em um novo programa: *Google ARCore Services* (GOOGLE..., 2021). Sua proposta é que uma biblioteca seja instalada no celular para que os aplicativos tenham acesso, trazendo a vantagem da redução do tamanho das aplicações produzidas. No entanto, somente uma lista restrita de *smartphones* modernos podem instalar essa biblioteca, a justificativa dos desenvolvedores é a compatibilidade com o sistema (ARCORE..., 2021).

Seguido disso, durante a pesquisa foi encontrado uma nova ideia *open-source*, também do *Google*, chamado *Mediapipe*, que tinha a proposta de entregar muitas ferramentas de visão computacional com ML (*machine learning*) integrado (MEDIAPIPE..., 2022). O projeto foi iniciado em julho de 2019, e tem ganhado mais popularidade por ser gratuito, multi-plataforma e facilitar muito a aplicação de ML para *Android*, *iOS*, e *PC*. No momento da descoberta, não era conhecido a compatibilidade do *Mediapipe* com *API* antigas de *Android* e muito menos o seu comportamento em óculos de realidade aumentada, por isso, a ideia foi reservada.

O que podemos concluir do desenvolvimento de aplicações em realidade aumentada com *Android Studio* é que mesmo existindo ferramentas robustas de criação de *apps* em AR, eles são restritos às novas versões de *Android*, que por sua vez, estão presentes somente em dispositivos de nova geração, i.e., de lançamento posteriores a 2018. Por fim, a incompatibilidade das bibliotecas com o sistema e a construção dos óculos AR foram os motivos para a procura de um novo ambiente de desenvolvimento, que seja mais versátil e comporte bem com os recursos do *Moverio BT-350*.

2.3 Estudos de desenvolvimento Unity

O programa apresentado no capítulo "Moverio AR" foi o único programa em *Unity* disponibilizado na documentação dos óculos, o projeto foi elaborado no *Unity* 2017, mesmo com alguns alertas de incompatibilidade, o projeto pôde ser compilado com êxito e funcionou nos óculos. Novamente, a falta do suporte da *EPSON* deixou incerto se era uma boa opção construir um aplicativo somente baseado na documentação disponível. Por isso, iniciamos novamente uma pesquisa sobre as ferramentas para suporte de AR nessa plataforma.

O *Vuforia* foi o primeiro *plugin* a ser testado no ambiente do *Unity* (VUFORIA..., 2021). Atualmente, ele apresenta mais opções rastreio para AR, mas no momento em que foi experimentado pela pesquisa as opções eram de usar imagens e algumas formas 3D como marcadores para suas projeções. Foi testado a detecção de imagem com um pedaço da logo da *Coca-cola™* como alvo, a figura 2.6 ilustra os resultados.



Figura 2.6: Testes com diferentes modelos de diferentes complexidades, constatando performance acima de 15 quadros por segundo em todos os casos. Fonte: Autor.

As formas de detecção tridimensionais não eram compatíveis com a detecção da posição da cabeça de um paciente, por isso esse recurso não foi testado. O *Vuforia* é uma ferramenta que

trabalha com *Android* de *API 23* ou superior, infelizmente, também incompatível com o *Moverio BT-350* que possui uma *API 22* e sem chances de atualização. Outras ferramentas foram encontradas como o *Wikitude™* que também era para *API 23* (WIKITUDE..., 2022), porém uma outra ferramenta denominada *ARFoundation* que estima a posição da face humana chamou a atenção por termos a oportunidade de criar uma demonstração da ideia final do projeto nos óculos (ARFOUNDATION..., 2022).



Figura 2.7: A primeira imagem foi o primeiro teste com a projeção de um cubo na região da testa. A segunda detalha a superfície estimada da face e a projeção do cérebro. Fonte: Autor.

A instalação do *ARFoundation* foi feita no *Unity 2018* e apresentou compatibilidade com celulares de *API 28* ou superior. Os testes consistiram em utilizar a posição da face encontrada pelo *plugin* para projetar um objeto em referência desses dados. A aplicação teve um ótimo resultado e foi possível apresentar o objetivo principal para professores e médicos da área pela aproximação visual dos testes com o resultado final esperado, visto na figura 2.7. Nesse momento, criamos um nome fantasia para facilitar as apresentações do projeto: *VCraniun*.

Retornando ao objetivo prático de encontrar um meio de desenvolver o projeto, concluímos que temos muitas opções disponíveis de desenvolvimento, mas sem muito suporte para o *Moverio BT-350*. O *Unity* foi escolhido para seguir a pesquisa pela sua facilidade de elaboração de cenários tridimensionais e compatibilidade com o sistema de outros óculos de AR do mercado. Dessa maneira, podemos garantir que os próximos trabalhos que serão desenvolvidos, não dependem do equipamento que temos hoje, i.e., se futuramente precisarmos trocar para um óculos mais moderno, uma adaptação será feita e aproveitaremos o material produzido até o momento.

Dado o histórico de testes do projeto, foi decidido prosseguir sem depender de *plugins* e bibliotecas *closed-source*, isso garantirá um maior controle do *software* elaborado e também

exigirá um estudo mais aprofundado de como os algoritmos de visão computacional funcionam. Essa mudança trará um contato maior com problemas discretizados de programação, auxiliando o processo do estudo e apresentação de dúvidas em seminário para os estudantes e professores dessa área de pesquisa na universidade.

2.4 Revisão bibliográfica

Aqui podemos analisar com mais atenção os artigos da literatura e suas diferentes soluções para sistemas de realidade aumentada aplicados em cirurgias. O trabalho que mais auxiliou a pesquisa foi uma revisão literária de aplicações em AR para neurocirurgias: "*Enhancing Reality: A Systematic Review of Augmented Reality in Neuronavigation and Education*" (CHO et al., 2020). Esse artigo faz uma breve apresentação da aplicabilidade de AR em ambiente cirúrgico e tabela as informações de doze pesquisas mostrando as patologias tratadas; descrição de método; e resultado da precisão da projeção em AR.

Dentre as pesquisas descritas estava o artigo de Maruyama: *Smart Glasses for Neurosurgical Navigation by Augmented Reality* (MARUYAMA et al., 2018). Sua equipe de pesquisadores construíram um sistema que utilizava os óculos *Moverio BT-200*, um modelo que se assemelha muito com o que possuímos no laboratório, para auxiliar a visualização de tumores cerebrais em dois pacientes, além disso, foi também possível exibir a escala; o crânio; e os vasos da superfície do cérebro nos óculos.



Figura 2.8: (A) Duas câmeras para detectar movimento, (B) Marcadores para paciente, (C) Marcadores nos óculos, (D) Visualização nos óculos. Fonte: (MARUYAMA et al., 2018).

O método que o artigo utilizou apresentou a dinâmica da integração de componentes para uma arquitetura de um sistema AR: Empregar câmeras estereoscópicas para a detecção de

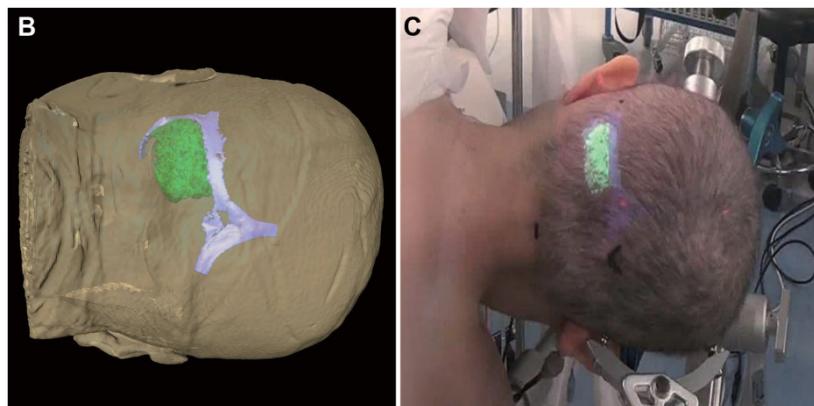


Figura 2.9: (B) Computação gráfica da reconstrução do cérebro do paciente, (C) Visualização em AR antes da incisão. Fonte: (MARUYAMA et al., 2018).

marcadores nos óculos e na cabeça do paciente; relacionar com os dados da reconstrução 3D do cérebro; utilizar parâmetros da calibração; e fazer a exibição nos óculos. Por meio desse sistema, o resultado obtido é visto na figura 2.9.

A precisão da projeção, segundo o artigo, foi de $2.1 \pm 1.3 \text{ mm}$ com a mediana de 1.8 e a remoção total de 5 tumores com sucesso e sem complicações pós-operatórias. Além disso, o sistema tem uma interface de fácil utilização e conveniência para o médico, sendo possível desabilitar a projeção em qualquer momento da cirurgia, teve um custo menor que os sistemas de neuronavegação convencionais e, por fim, pode ser instalado em outros óculos de AR comerciais disponíveis, portanto, não se limitando ao seu modelo do *Moverio* (MARUYAMA et al., 2018).

Os resultados apresentados por essa pesquisa e que a conjectura apresentada na seção anterior - aplicação baseado no *Unity* para a compatibilidade com mais óculos do mercado - foram confirmados possíveis. Contudo, antes de simplesmente seguir os passos de Maruyama, decidimos fazer uma listagem das arquiteturas utilizadas em outros sistemas para tentar ponderar suas características para optarmos pela opção que mais atende as necessidades da pesquisa.

2.5 Elaboração do VCranium

2.5.1 Arquitetura do Sistema

A definição da arquitetura do sistema é muito importante para configurar o funcionamento do sistema, estabelecer suas vantagens e desvantagens, e implicará como nós trabalhos

em seu desenvolvimento. Para isso, a equipe realizou reuniões e debates para estabelecer uma solução que seja compatível para um período de seis meses e respeitando as medidas de prevenção por afastamento pela pandemia de COVID-19. Ainda assim, precisamos de uma arquitetura versátil e que objetiva facilitar a mudança de métodos de estimativa a posição da projeção em AR.

Para tal, dividiremos o sistema em computador e óculos: O computador estima a posição da projeção e os óculos fazem a visualização nos olhos do usuário. Isso simplifica a mudança de método aplicado pelo computador, melhorando a recepção de novos algoritmos da equipe de estudos do laboratório. Esse invento foi aplicado no artigo de Maruyama, que utilizou um *software* de visualização médica e enviou os dados de exibição para o *Unity* nos óculos (MARUYAMA et al., 2018). Com a definição dos componentes da arquitetura, deve ser definido uma comunicação entre os dispositivos. Um exemplo da literatura foi o uso do computador para adquirir informações de sinais vitais de equipamentos médicos e o envio para os óculos via *TCP/IP* (Protocolo de Controle de Transmissão / Protocolo de *Internet*) (ARPAIA et al., 2021).

Utilizando o recurso de conexão *wireless* dos óculos, podemos nos conectar com o computador via *LAN* (Rede de área local). Na questão da oscilação, basta estar em um raio de cinco metros da origem do sinal, livre de obstáculos, que o sinal possui uma boa velocidade e latência, $v \pm u \text{ Mbps}$ e $s \pm e \text{ ms}$, respectivamente, e dessa maneira, temos liberdade de enviarmos dados grandes como imagens. Essa característica tem afinidade com a arquitetura mencionada anteriormente: O computador aplica algoritmos de visão computacional em imagens recebida dos óculos e retorna os resultados para os óculos novamente.

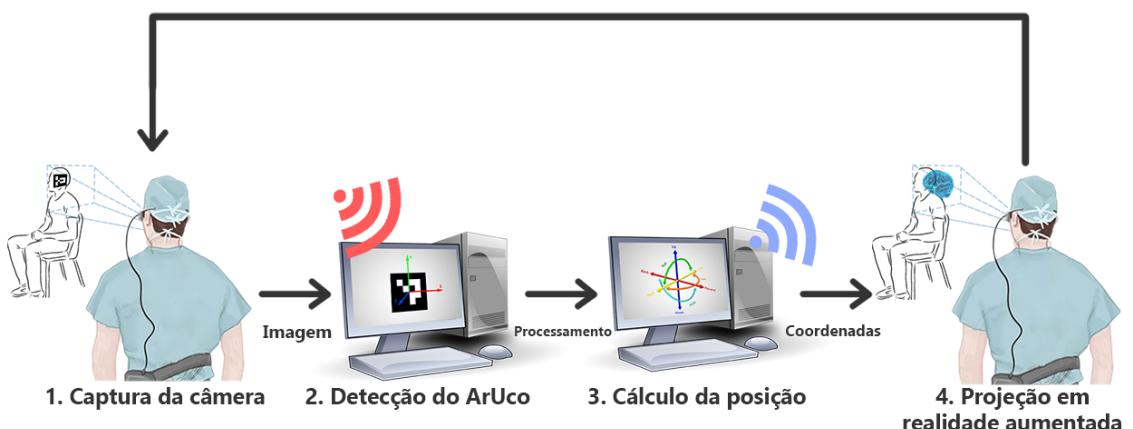
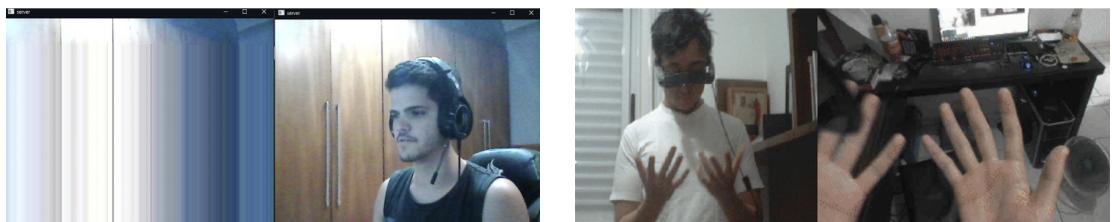


Figura 2.10: A figura representa o funcionamento do sistema. (1) Captura a imagem do paciente e envia para o computador. (2) Faz uma varredura na imagem e identifica o marcador ArUco. (3) Calcula a posição do marcador e envia as coordenadas para os óculos. (4) Recebe as informações e exibe a projeção para o usuário e então retorna para o passo 1. Fonte: Autor.

2.5.2 Comunicação

A parte prática da programação do sistema teve o desafio de conciliar a comunicação do computador programado em *Python* com os óculos programados em *Unity*. As pesquisas na *web* demonstraram essa comunicação via *TCP/IP* com a utilização de *network sockets* (YOUTUBE..., 2020). No entanto, o que precisava ser feito é a transmissão de vídeo em tempo real (*video streaming*), essa ocasião é diferente do envio de uma mensagem de texto, que está na ordem de centenas de *bytes*, uma única imagem pode carregar milhões de *bytes* de informação.

Nos testes iniciais, a tentativa foi realizar o processo contínuo de obturação de foto e envio dos dados via *TCP* ao computador. Porém, ocasionalmente, a visualização da imagem era cortada horizontalmente, parecendo que "faltou um pedaço". Isso ocorre pois no processo de recepção de dados do *socket*, a instrução de leitura não sabe o tamanho do arquivo a ser recebido da rede, i.e., o computador não saber o tamanho da imagem eventualmente causava a renderização parcial das informações da foto, resultando em uma imagem cortada.



(a) Comparação entre uma imagem defeituosa e (b) Êxito na transmissão em tempo real das imagens dos óculos para o computador.

Figura 2.11: Resultados dos testes de comunicação Fonte: Autor.

A solução desse problema foi fazer um envio de uma mensagem de tamanho fixo (*header*) avisando ao computador o tamanho da próxima imagem, garantindo a captura completa da informação de cada foto (Figura 2.11a). Esse estudo e investigação do problema ampliou muito os conhecimentos de redes e seus protocolos de comunicação. Foi possível adquirir um resultado melhor que o esperado na transmissão de vídeo, obtendo uma taxa superior a 20 quadros por segundo, sua variação é sensível ao conteúdo da imagem por causar variação na taxa de compressão do formato *JPEG* (*Joint Photographic Experts Group*).

2.5.3 Estimação da posição da projeção

A aplicação objetiva exibir os detalhes da anatomia cerebral de um paciente em AR para auxiliar os procedimentos operatórios de um neurocirurgião. Para isso, devemos utilizar um método que estime a relação entre o mundo real e virtual para sobrepormos a cabeça do paciente com a vista 3D dos seus dados da tomografia computadorizada. A procura desse método é o objeto de pesquisa de muitas referências bibliográficas. Nesse ponto da pesquisa, o projeto pode receber os métodos pesquisados por outros estudantes do laboratório, no entanto, optamos por um método com baixa complexidade para darmos ênfase no objetivo de completarmos todo o sistema e, em um futuro breve, torná-lo em algo mais adequado para o ambiente cirúrgico.

O método de estimação escolhido foi a detecção de marcadores fiduciais nativos da biblioteca do *OpenCV*, o *ArUco*. A estratégia de marcadores, especificamente os com forma quadrada, é comumente utilizada em aplicações de realidade aumentada pela sua rápida responsividade e robustez (RAMIREZ; SALINAS; CARNICER, 2018). Ele é especial por ter quatro pontos proeminentes (os quatro vértices) que facilitam sua detecção e seu conteúdo interno, preenchido com um padrão único, serve de identificador para o computador (POROYKOV et al., 2020).

O algoritmo da detecção *ArUco* foi implementado junto ao programa *Python* do computador. Esse módulo tem como entrada uma imagem dos óculos e saída as coordenadas da posição e rotação do marcador. Os dados são formatados em uma mensagem de tamanho fixo e enviado por *TCP*, semelhante ao *header* da imagem. Na rotina de execução dos óculos, ele primeiro obtura a imagem e envia para o computador, então ele espera a resposta com as informações do *ArUco* correspondentes à imagem e por fim ele posiciona a projeção AR. Na detecção de quaisquer erros de formatação dos *header*, são emitidos como erros fatais no programa, parando todos os processos da execução, vistos no fluxograma da figura 2.12.

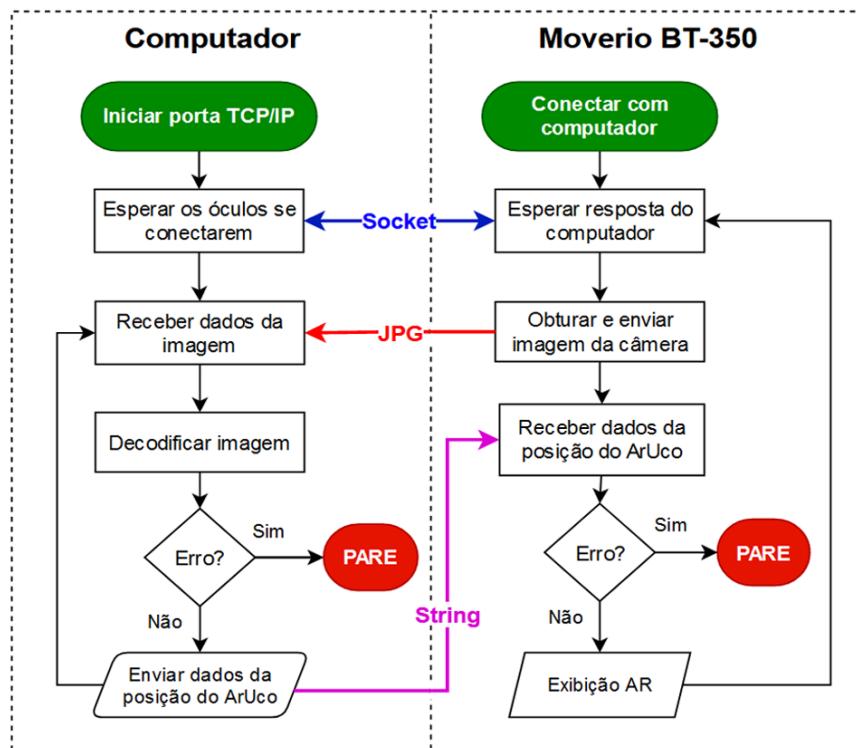


Figura 2.12: Fluxograma dos processos da execução do *VCranium*. Fonte: Autor.

2.5.4 User Interface

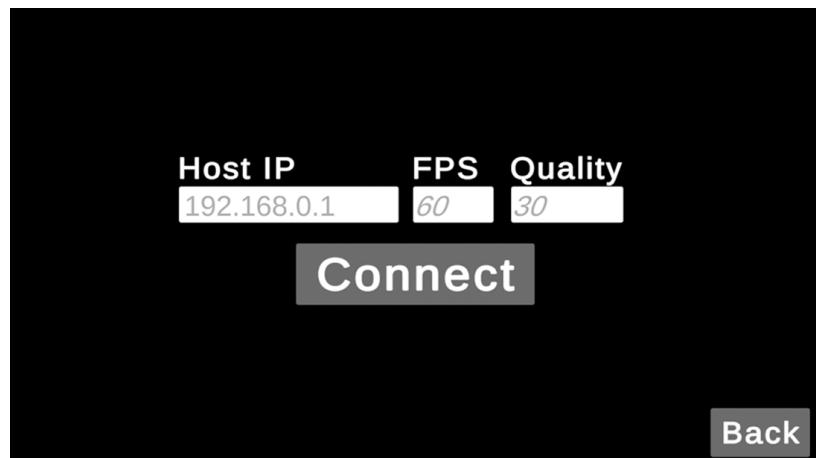


Figura 2.13: Interface de conexão com computador. Fonte: Autor.

Referências bibliográficas

- ABOUT OpenCV. [S.l.: s.n.]. <https://opencv.org/about/>. Acessado em 17 de fev. de 2020.
- ARCORE: supported device. [S.l.: s.n.], 2021. <https://developers.google.com/ar/devices>. Acessado em: 7 de agosto de 2021.
- ARFOUNDATION: Documentação do *plugin*. [S.l.: s.n.], 2022. <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html>. Acessado em: 26 de fevereiro de 2022.
- ARPAIA, P. et al. Metrology-Based Design of a Wearable Augmented Reality System for Monitoring Patient's Vitals in Real Time. **IEEE Sensors Journal**, v. 21, n. 9, p. 11176–11183, mai. 2021. ISSN 1530-437X. DOI: 10.1109/JSEN.2021.3059636. Disponível em: <<https://ieeexplore.ieee.org/document/9354808/>>.
- CHO, J. et al. Enhancing Reality: A Systematic Review of Augmented Reality in Neuronavigation and Education. **World Neurosurgery**, p. 186–195, mai. 2020. DOI: 10.1016/j.wneu.2020.04.043.
- GOOGLE Play: Google Play Services para RA. [S.l.: s.n.], 2021. <https://play.google.com/store/apps/details?id=com.google.ar.core>. Acessado em: 5 de agosto de 2021.
- MARUYAMA, K. et al. Smart Glasses for Neurosurgical Navigation by Augmented Reality. **Operative Neurosurgery**, v. 15, n. 5, p. 551–556, nov. 2018. ISSN 23324260. DOI: 10.1093/ons/oxz279. Disponível em: <<https://academic.oup.com/ons/article/15/5/551/4823484>>.
- MEDIAPIPE: *README* do projeto no *Github*. [S.l.: s.n.], 2022. <https://github.io/mediapipe/>. Acessado em: 26 de fevereiro de 2022.
- POROYKOV, A.; KALUGIN, P.; SHITOV, S.; LAPITSKAYA, I. Modeling ArUco Markers Images for Accuracy Analysis of Their 3D Pose Estimation. **Proceedings of the 30th International Conference on Computer Graphics and Machine Vision (GraphiCon 2020). Part 2**, short14–1–short14–7, dez. 2020. DOI: 10.51130/graphicon-2020-2-4-14. Disponível em: <<http://ceur-ws.org/Vol-2744/short14.pdf>>.

RAMIREZ, F. J. R.; SALINAS, R. M.; CARNICER, R. M. Speeded up detection of squared fiducial markers. **Image and Vision Computing**, v. 76, June, p. 38–47, 2018. ISSN 02628856. DOI: 10.1016/j.imavis.2018.05.004.

SCENEFORM: Quickstart for Android. [S.l.: s.n.], 2021. <https://developers.google.com/sceneform/develop/android-quickstart>. Acessado em: 5 de agosto de 2021.

SERVÍCIO de Neurocirurgia retira lesão cerebral com paciente acordado: Youtube. [S.l.: s.n.], 2019. <https://youtu.be/-n1x25yu04w>. Acessado em: 5 de março de 2022.

UDEMY: Desenvolvimento *Android* Completo 2021. [S.l.: s.n.], 2021. <https://www.udemy.com/course/curso-de-desenvolvimento-android-oreo/>. Acessado em: 5 de agosto de 2021.

UNITY: Arquivo de download Unity. [S.l.: s.n.], 2021. <https://unity3d.com/pt/get-unity/download/archive>. Acessado em: 10 de agosto de 2021.

VUFORIA: Developer Portal. [S.l.: s.n.], 2021. <https://developer.vuforia.com>. Acessado em: 9 de agosto de 2021.

WIKITUDE: Augmented Reality. [S.l.: s.n.], 2022. <https://www.wikitude.com/>. Acessado em: 1 de março de 2022.

YOUTUBE: Connect Csharp and Python tutorial. continuous communication. [S.l.: s.n.], 2020. <https://youtu.be/veIIiU9qTsGI>. Acessado em: 8 de março de 2022.