

APLICABILIDADE DO SENSOR LIDAR NA DETECÇÃO DE AMBIENTES E OBJETOS PARA ORIENTAÇÃO DE PESSOAS COM DEFICIÊNCIAS VISUAIS

Bruno Henrique de Borba, Dalton Solano dos Santos – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

bhborba@furb.br, dalton@furb.br

Resumo: Este trabalho apresenta o desenvolvimento de uma aplicação que utiliza a tecnologia do sensor LiDAR para auxiliar pessoas com alguma deficiência visual a se locomoverem em ambientes internos, evitando colisões com obstáculos, detectando objetos e orientando o usuário de acordo com suas respectivas posições, sem necessidade de conexão com a internet ou servidores dedicados para realizar as etapas de detecção e processamento. Para isso, foram utilizados frameworks de realidade aumentada da Apple como o ARKit e Reality Kit, bem como frameworks para detecção de objetos como o Vision e o CoreML. Obtiveram-se resultados satisfatórios para evitar obstáculos e detectar objetos, e para a orientação do usuário a localizar os objetos detectados pela aplicação. Apesar dos resultados, existem imprecisões que não permitem que a aplicação seja utilizada pelo usuário final de forma segura.

Palavras-chave: Realidade aumentada. ARKit. LiDAR. Detecção de objetos.



1 INTRODUÇÃO

De acordo com a Organização Mundial da Saúde, em 2021 pelo menos 2,2 bilhões de pessoas tem algum tipo de deficiência visual para perto ou para longe. De todas as 2,2 bilhões de pessoas citadas, pelo menos 1 bilhão de casos poderiam ter sido prevenidos ou ainda não foram tratados. Grande parte dos casos de deficiência visual e cegueira atingem pessoas com idade acima dos 50 anos, mas podem acontecer em todas as idades (WORLD HEALTH ORGANIZATION, 2021).

A Classificação Internacional de Doenças (International Classification of Diseases - IDC) separa a deficiência visual em: para longe, sendo estas de níveis leve, moderada, severa e cegueira; e para perto, caracterizada por visão inferior a N6 ou M0.8 (valores de acuidade visual) com correção existente (WORLD HEALTH ORGANIZATION, 2020). A deficiência visual pode ocorrer devido a alguma doença, como catarata, glaucoma, opacidade da córnea, retinopatia diabética, tracoma e até erro refrativo não resolvido (BOURNE, 2021). No geral, pessoas com deficiência visual podem sofrer mais com outros problemas quando comparados com outras pessoas, como com a solidão (BRUNES; HANSEN; HEIR, 2019) e depressão (RENAUD; BÉDARD, 2013).

Outro aspecto importante a ser destacado são as dificuldades na execução de tarefas diárias em ambientes fechados, sendo as principais: encontrar um quarto pelo número, encontrar um elevador, ler números em um banco, encontrar escadas e reconhecimento de objetos (PLIKYNAS *et al.*, 2020). Para a solução de problemas como estes, existem sistemas que se propõem a auxiliar na detecção de objetos e ambientes, mas que contam com a necessidade de um servidor para processar o reconhecimento (JIANG; LIN; QU, 2016), de câmeras ou equipamentos acoplados no corpo do usuário (PHAM; LE; VUILLERME, 2016), ou que precisam ser vestíveis e utilizados em conjunto com outros equipamentos (BAI *et al.*, 2019).

Com o advento dos dispositivos móveis e o aumento da sua capacidade de processamento constante (APPLE, 2021e), novos hardwares são adicionados aos dispositivos. É o caso do sensor Light Detection and Ranging (LiDAR), que é capaz de coletar uma nuvem de pontos com coordenadas x, y e z de objetos ao redor do ambiente. Um dos grandes benefícios do sensor é não ter influência da condição de luz, podendo trabalhar em dias e noites por exemplo (WU, 2018).

Diante das informações apresentadas, esse trabalho implementou um aplicativo móvel que tem por objetivo utilizar a tecnologia do sensor LiDAR para auxiliar pessoas com algum tipo de deficiência visual a se locomoverem em ambientes fechados, detectando objetos e indicando suas direções, bem como alertando e desviando o usuário de possíveis colisões com obstáculos. Os testes da aplicação consistiram em validar se a detecção dos objetos era precisa, bem como as orientações fornecidas para encontrá-los, e se a detecção de colisões é consistente com a realidade.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são descritas as tecnologias e ferramentas utilizadas para o desenvolvimento desta aplicação. Serão apresentados os seguintes assuntos: Sensor LiDAR, frameworks ARKit e Reality Kit para trabalhar com realidade aumentada, e frameworks Core ML e Vision para aprendizagem de máquina e detecção de objetos respectivamente.

2.1 SENSOR LIDAR

O sensor LiDAR é capaz de detectar uma nuvem de pontos com as coordenadas x, y e z de objetos ao seu redor (WU, 2018). Utilizado comumente em carros autônomos, o sensor fornece informações valiosas em condições que outros sensores não são capazes de fornecer, sendo seu principal foco a informação de distância, medindo o tempo de “viagem” da luz emitida (EUROSENSORS CONFERENCE, 2018).

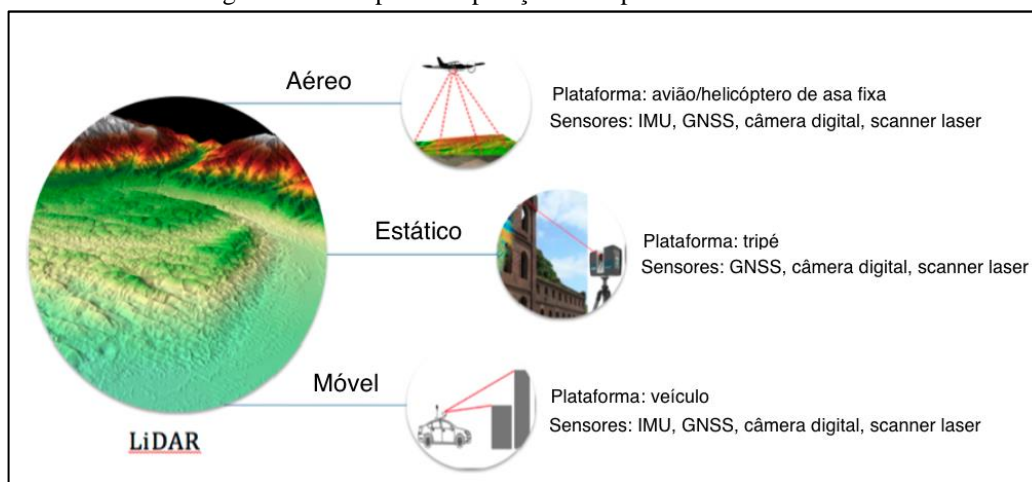
A forma de capturar os dados difere pela posição do sensor, ou seja, com o LiDAR sendo baseado no ar ou no solo. Ambos os mecanismos variam na forma de adquirir os dados, podendo ser por varredura, precisão e resolução, mas possuem diversas similaridades. Uma das semelhanças é que ambos os sistemas podem capturar dados através de uma nuvem de pontos e simultaneamente adquirir as imagens (MUHADI *et al.*, 2020).

O LiDAR baseado no ar é um sistema multissensorial que consiste em vários componentes, sendo estes plataforma, scanner a laser, hardware de posicionamento, equipamento fotográfico ou de gravação de vídeo, computador e armazenamento de dados. No LiDAR aerotransportado, a plataforma para montar o scanner a laser pode ser uma aeronave de asa fixa ou um helicóptero, que é usado para sobrevoar com o sensor sobre uma região de interesse (MUHADI *et al.*, 2020).

Já o LiDAR terrestre ou baseado no solo, é uma versão terrestre do LiDAR aerotransportado, frequentemente usado para mapeamento topográfico e de terreno. O LiDAR terrestre inclui varredura a laser estacionária, em que o sensor é montado em um tripé para posições fixas, e varredura laser móvel, em que o sensor é montado em uma plataforma móvel baseada no solo, como em um veículo (MUHADI *et al.*, 2020).

O scanner a laser móvel tem modos de coleta de dados similares ao LiDAR aerotransportado. Requer apenas uma direção de escaneamento, onde as demais são realizadas pela plataforma móvel. Em sistemas de laser móveis, o scanner a laser é montado em veículos que se locomovem, como carros e vans. Pelo movimento contínuo, tecnologias baseadas em posicionamento como a Unidade de Medida Inercial (IMU - Inertial Measurement Unit) ou o Sistema de Posicionamento Global (Global Positioning System – GPS) são necessárias para medir com precisão as respectivas posições e orientações (MUHADI *et al.*, 2020). A Figura 1 demonstra os princípios de operação dos tipos de scanner a laser.

Figura 1 - Princípios de operação dos tipos de scanner a laser



Fonte: Muhadi *et al.* (2020).

Inicialmente o LiDAR foi desenvolvido com o objetivo de mapear áreas aeronáuticas, como florestas, mantos de gelo, oceanos e atmosfera, e áreas aeroespaciais, como a superfície da lua na missão Apollo 15 (PETIT, 2020). Somente após a disponibilidade do GPS e das IMUs no final dos anos 1980, foi possível obter uma maior precisão dos dados do LiDAR (GREGERSEN, 2016).

Atualmente o LiDAR é aplicado nas mais diversas áreas, como por exemplo na agricultura para análise da qualidade do solo, veículos autônomos para percepção do mundo ao redor, arqueologia para definir variação topográfica, e na legislação para conduzir análises forenses (USAU, 2020). Áreas como infraestrutura, cidades inteligentes, internet das coisas, transportes, logísticas e aplicações industriais certamente irão, ou já estão sendo altamente beneficiadas com as tecnologias do LiDAR (PETIT, 2020).

2.2 FRAMEWORK ARKIT

Com a evolução da computação, surgem novos métodos de interação do usuário com o computador e apresentação de dados. Tais métodos incluem Realidade Aumentada e Realidade Virtual. Ambos os métodos de apresentação de dados diferem principalmente na proporção de imagens geradas por computador para o mundo real. De

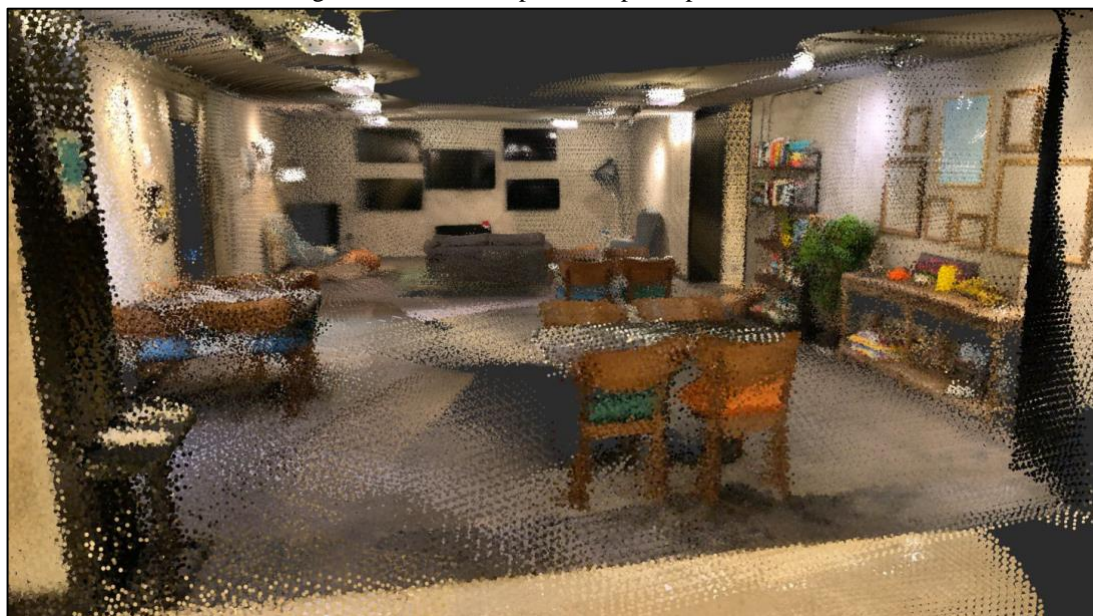
forma resumida, Realidade Aumentada é quando o mundo real é enriquecido com dados gerados pelo computador, e Realidade Virtual ocorre quando os dados gerados por computador obscurecem completamente o mundo real (NOWACKI; WODA, 2019).

Graças ao rápido desenvolvimento dos dispositivos móveis, milhões de usuários tem acesso a dispositivos móveis poderosos, capazes de lidar com Realidade Aumentada e Realidade Virtual quase em tempo real. Em 2017, Apple e Google apresentaram duas interfaces de programação competitivas que suportam a criação de aplicativos de Realidade Aumentada para dispositivos móveis: ARKit, lançado em 19 de setembro de 2017, e ARCore, com versão estável lançada em 1 de março de 2018 (NOWACKI; WODA, 2019).

ARKit é um *framework* desenvolvido pela Apple, focado em Realidade Aumentada. O ARKit combina rastreamento de movimento do dispositivo, captura de cena de câmera, processamento avançado de cena e conveniências de exibição, facilitando a construção de uma experiência de realidade aumentada ao utilizar a câmera frontal ou traseira de um dispositivo iOS (APPLE, 2021b).

A versão 4 do ARKit lançada em 2020, introduziu uma nova Interface de Programação de Aplicativos (Application Programming Interface – API) com foco no sensor LiDAR, sendo dessa forma possível acessar informações detalhadas de profundidade geradas pelo sensor. A geometria de cena do ARKit permite criar um mapa topológico de algum espaço com rótulos, identificando pisos, paredes, tetos, janelas, portas e assentos, o que também contribui para a coleta de mais informações necessárias em trabalhos de realidade aumentada (APPLE, 2021d). A Figura 2 demonstra a nuvem de pontos que o LiDAR captou em um ambiente interno com o auxílio do ARKit.

Figura 2 - Nuvem de pontos captada pelo LiDAR



Fonte: Everypoint (2020).

Para visualizar e interagir com uma cena reconstruída, o ARKit usa o sensor LiDAR para criar um modelo poligonal do ambiente físico. Dessa forma, o sensor obtém as informações de profundidade de uma grande área na frente do usuário, e o ARKit a converte em uma série de vértices que se conectam para formar uma espécie de malha (APPLE, 2021h). A malha gerada possibilita uma maior identidade ao mundo detectado pelo sensor, sendo possível inclusive gerar física nos objetos do mundo real, e criar colisões com outros objetos virtuais.

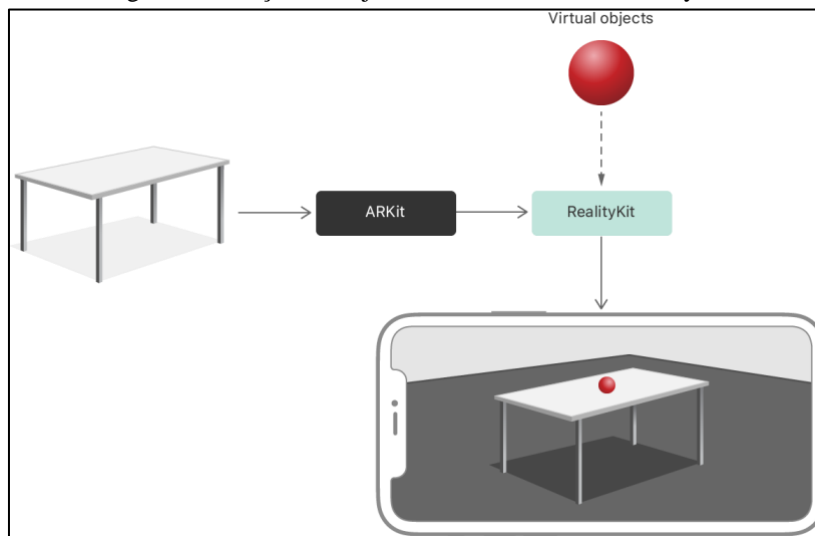


FRAMEWORK REALITYKIT

Enquanto o ARKit trabalha diretamente com os dados obtidos por sensores como o LiDAR, o *framework* RealityKit atua na implementação e renderização de simulações em terceira dimensão nas aplicações iOS (LARSSON; RUNESSON, 2021). Este *framework* foi lançado em 2019, construído especialmente para realidade aumentada que utilize renderização foto-realística, efeitos de câmera, animações, física e mais (APPLE, 2021g).

Quando utilizado em conjunto com as informações do sensor LiDAR, o RealityKit permite que objetos virtuais interajam com os arredores do ambiente físico capturado pela câmera. Desta forma, objetos podem ser colocados embaixo de mesas, atrás de paredes, ou em esquinas que possibilitam visualizar apenas partes do objeto virtual como esperado em um ambiente físico (APPLE, 2021g). A Figura 3 ilustra a adição de um objeto virtual em um ambiente capturado pela câmera com o RealityKit.

Figura 3 – Adição de objeto virtual utilizando o RealityKit



Fonte: Apple (2021f).

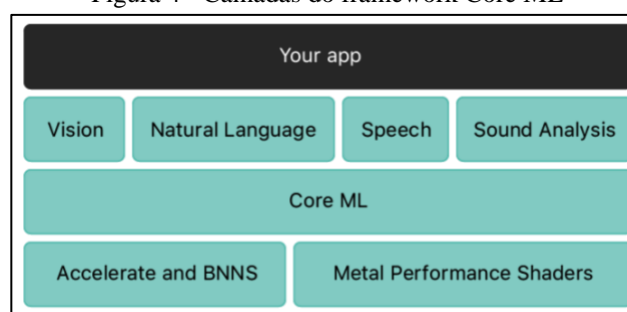
Para adicionar objetos na cena, utiliza-se a classe *AnchorEntity*. Com essa âncora, o RealityKit consegue colocar o objeto de acordo com a percepção do mundo físico criada, como colocar o objeto em cima ou abaixo da mesa por exemplo. Com a âncora criada, basta adicioná-la à coleção de âncoras da cena, e o RealityKit conseguirá mover todas as âncoras de acordo com cada atualização na cena, sem a necessidade de que isso esteja explicitamente programado em código (APPLE, 2021a).

2.4 FRAMEWORK CORE ML E FRAMEWORK VISION

O Core ML é um *framework* desenvolvido pela Apple que permite integrar modelos de aprendizagem de máquina em aplicativos compatíveis com macOS, iOS, iPadOS, tvOS e watchOS. Um modelo é o resultado da aplicação de um algoritmo de aprendizado de máquina a um conjunto de dados de treinamento, sendo utilizado para realizar previsões baseadas em novos dados de entrada (APPLE, 2021c).

A utilização do Core ML permite que o processamento do modelo aconteça estritamente no dispositivo do usuário, removendo a necessidade da utilização de conexão com a internet, o que ajuda a manter os dados do usuário privados e o aplicativo responsivo. Com o uso do Core ML é possível realizar tarefas como análise de imagens utilizando o *framework* Vision, processamento de texto com o *framework* Natural Language, o *framework* Speech para conversão de áudio para texto, e o *framework* Sound Analysis para identificar sons em áudio (APPLE, 2021c). A Figura 4 demonstra as camadas do Core ML.

Figura 4 - Camadas do framework Core ML



Fonte: Apple (2021c).

Conforme citado anteriormente, o *framework* Vision realiza análise de imagens, sendo possível detectar rostos e pontos de referência de rostos, detecção de textos, reconhecimento de código de barras, registro de imagens e rastreamento de recursos gerais. Além dessas funcionalidades nativas, o Vision também permite a utilização de modelos personalizados do Core ML para tarefas como classificação ou detecção de objetos.

2.5 TRABALHOS CORRELATOS

Foram selecionados três trabalhos correlatos que possuem características semelhantes aos objetivos do estudo desenvolvido. O primeiro trabalho é um sistema para reconhecimento em tempo real de objetos, com seus resultados convertidos em um som tridimensional (JIANG; LIN; QU, 2016), descrito no Quadro 1. No Quadro 2 é apresentado o trabalho de detecção em tempo real de objetos e obstáculos utilizando o Microsoft Kinect (PHAM; LE; VUILLERME,

2016). No Quadro 3 é apresentado um dispositivo vestível capaz de detectar obstáculos e objetos, e orientar o usuário em ambientes internos e externos (BAI *et al.*, 2019).

Quadro 1 – Let Blind People See: Real-Time Visual Recognition with Results Converted To 3D Audio

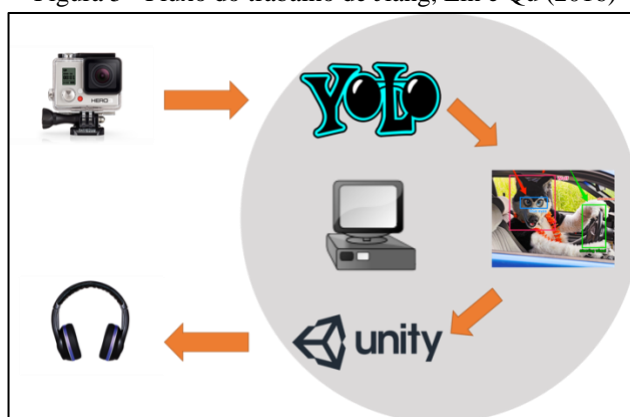
Referência	Jiang, Lin e Qu (2016)
Objetivos	Permitir que pessoas com deficiência visual possam localizar objetos em ambientes internos.
Principais funcionalidades	Reconhecimento de objetos e conversão dos seus resultados para um som tridimensional, orientando o usuário de sua localização.
Ferramentas de desenvolvimento	Unity para gerar o áudio tridimensional, algoritmo YOLO para reconhecimento de objetos.
Resultados e conclusões	A detecção de objetos teve um bom desempenho, e a utilização da descrição dos objetos por áudio tridimensional foi de grande auxílio para a localização e guia do usuário no ambiente.

Fonte: elaborado pelo autor.

O trabalho descrito por Jiang, Lin e Qu (2016) consiste em permitir que pessoas com deficiência visual tenham a possibilidade de reconhecer o ambiente ao seu redor a partir da detecção de objetos em tempo real, e de sua descrição por áudio para o usuário. A técnica utiliza a tecnologia denominada de simulação de som binaural, que facilita a identificação da direção onde o objeto se encontra. O sistema foi composto pela captura de vídeo feita por uma câmera GoPro, e as imagens foram transmitidas para um servidor, com o objetivo de realizar o reconhecimento da imagem em tempo real com modelos de detecção de objetos existentes, como o YOLO. Para o áudio tridimensional, utilizou-se o motor de jogos Unity. O som foi transmitido para o usuário por meio de fones de ouvido sem fio.

As imagens são transmitidas para um servidor que realiza a detecção dos objetos. A transmissão do vídeo em alta definição é feita em até 1 milissegundo, e o algoritmo do YOLO processa um único frame da imagem numa velocidade de 4 a 60 frames por segundo, dependendo do tamanho da imagem enviada. Nos testes realizados, constatou-se que a utilização da descrição dos objetos por áudio tridimensional foi de grande auxílio para a localização e guia do usuário no ambiente, mas alguns ajustes precisariam ser feitos, como por exemplo diminuir a frequência da repetição de objetos já detectados no ambiente (JIANG; LIN; QU, 2016). A Figura 5 demonstra o fluxo do trabalho proposto.

Figura 5 - Fluxo do trabalho de Jiang, Lin e Qu (2016)



Fonte: Jiang, Lin e Qu (2016).

Quadro 2 - Real-Time Obstacle Detection System in Indoor Environment For The Visually Impaired Using Microsoft Kinect Sensor

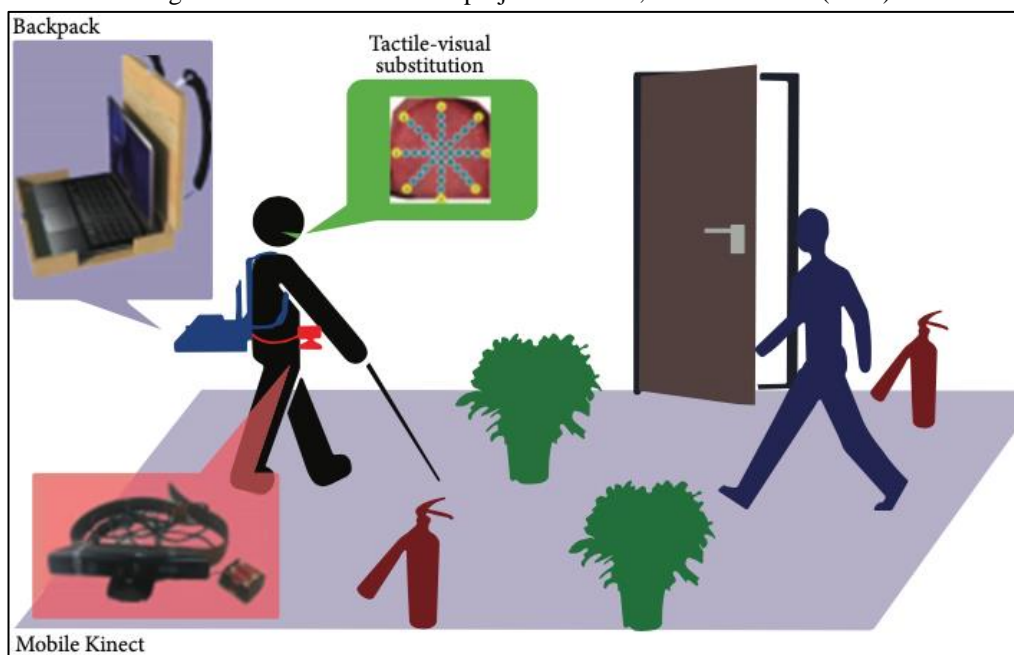
Referência	Pham, Le e Vuillerme (2016)
Objetivos	Detectar obstáculos em ambientes internos para orientar pessoas com deficiência visual utilizando o Microsoft Kinect.
Principais funcionalidades	Deteção de obstáculos no chão, paredes e escadas, e indicar ao usuário sua localização.
Ferramentas de desenvolvimento	Desenvolvido em C++ utilizando Visual Studio 2010
Resultados e conclusões	O sistema é capaz de detectar com acurácia paredes, portas e escadas. Houve problemas de acurácia em ambientes com muita luz, e algumas melhorias podem ser feitas para lidar com cenários como esses. De forma geral, o trabalho atinge o objetivo proposto.

Fonte: elaborado pelo autor.

O projeto de Pham, Le e Vuillerme (2016) foca em utilizar um método baseado no Microsoft Kinect para detectar obstáculos em ambientes internos. Este trabalho utiliza uma imagem tridimensional processada com informações de profundidade de cor, contando com o auxílio de um sensor Tongue Display Unit (TDU) para alertar o usuário. Desta

forma, é possível auxiliar pessoas com alguma deficiência visual, informando dos obstáculos ao redor do usuário, como escadas, paredes, portas e objetos indefinidos no chão. A Figura 6 demonstra o funcionamento do projeto.

Figura 6 - Funcionamento do projeto de Pham, Le e Vuillerme (2016)



Fonte: Pham, Le e Vuillerme (2016).

Quadro 3 - Wearable Travel Aid For Environment Perception And Navigation Of Visually Impaired People

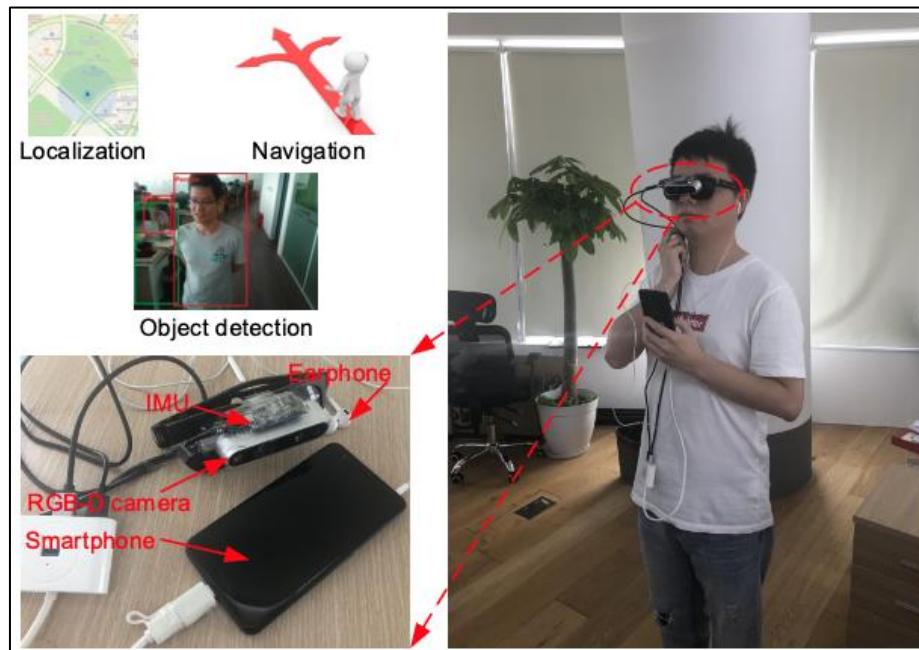
Referência	Bai <i>et al.</i> (2019)
Objetivos	Auxiliar na percepção e navegação de ambientes para pessoas com deficiência visual.
Principais funcionalidades	Deteccção de objetos e obstáculos, auxiliar na navegação do usuário em ambientes externos e internos.
Ferramentas de desenvolvimento	Não foi encontrado. Para a deteção de objetos, foi utilizada a Rede Neural Convolucional PeleeNet. Para o sistema de navegação em ambientes internos, foi utilizada a técnica Visual Simultaneous Localization (VSLAM).
Resultados e conclusões	O trabalho atinge o objetivo de orientar pessoas com deficiência visual para desviar de obstáculos, bem como detectar e orientar a direção de objetos. A navegação do usuário em ambientes internos e externos também atinge bons resultados.

O trabalho de Bai *et al.* (2019), tem como principais objetivos orientar rapidamente e com segurança o usuário em ambientes que este não tenha familiaridade, bem como reconhecer objetos em ambientes internos e externos. Para isto, propõe a utilização de um par de óculos com uma câmera RGB-D, uma Unidade de Medição Inercial (Inertial Measurement Units - IMU) e um smartphone.

Um sistema de reconhecimento de objetos baseados em uma Rede Neural Convolucional (Convolutional Neural Network - CNN) executada no smartphone é utilizada para aumentar a habilidade de percepção do usuário e promover o sistema de navegação, com informações semânticas dos arredores, como localizações e orientações dos objetos. Como interação humana para o alerta do usuário, é utilizado um som de bipe para obstáculos, reconhecimento de voz para entender comandos do usuário, e síntese de fala para dar informações semânticas dos objetos nos arredores do ambiente (BAI *et al.*, 2019).

O trabalho de Bai *et al.* (2019) é separado em aquisição de dados, localização, global path planning, evitar obstáculos, deteção de objetos e Interação Humano-Máquina (Human Machine Interaction - HMI). As duas principais capacidades do projeto são navegação e reconhecimento, sendo esse possível inclusive a utilização em ambientes externos. Mais detalhes do trabalho são apresentados na Figura 7.

Figura 7 - Trabalho proposto por Bai *et al.* (2019).



Fonte: Bai *et al.* (2019).

Para auxiliar na navegação do usuário, o trabalho de Bai *et al.* (2019) também utiliza mapas, sendo de serviços como GoogleMaps, QMap ou BaiduMap para ambientes externos, e a técnica de Visual Simultaneous Localization (VSLAM) para ambientes internos. Desta forma, é possível utilizar o algoritmo de Global Path Planning para montar os caminhos desejados pelo usuário.

3 DESCRIÇÃO DO APLICATIVO

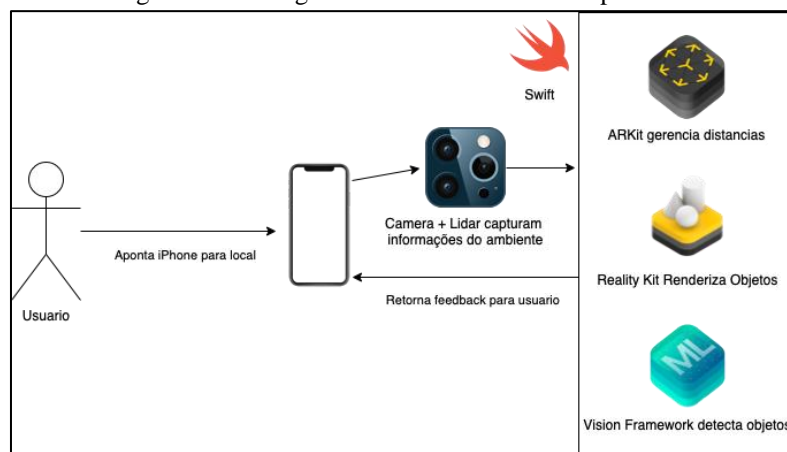
Nesta seção são apresentados os aspectos gerais do funcionamento e desenvolvimento do aplicativo. Para isso, são apresentadas 5 subseções. A primeira apresenta o funcionamento geral da aplicação. A segunda descreve como são detectadas e informadas ao usuário as colisões pelo aplicativo. A terceira demonstra como o aplicativo gera uma malha no ambiente detectado, e qual a sua importância para a aplicação. A quarta e quinta descrevem a forma utilizada para a detecção de objetos, e como são obtidas as direções necessárias para chegar ao objeto detectado respectivamente.

3.1 ESPECIFICAÇÃO

Para o desenvolvimento do aplicativo utilizou-se a linguagem de programação Swift. A tratativa das distâncias detectadas pelo sensor LiDAR foi feita pelo *framework* ARKit, a renderização de objetos pelo *framework* RealityKit, e a detecção dos objetos pelo *framework* Vision. O objetivo é que o usuário aponte o celular com a câmera direcionada para o ambiente onde deve acontecer o reconhecimento, e receba retornos referentes à objetos detectados e suas distâncias, bem como possíveis colisões com algo que esteja a sua frente. A Figura 8 apresenta uma visão geral do funcionamento do aplicativo.



Figura 8 - Visão geral do funcionamento do aplicativo



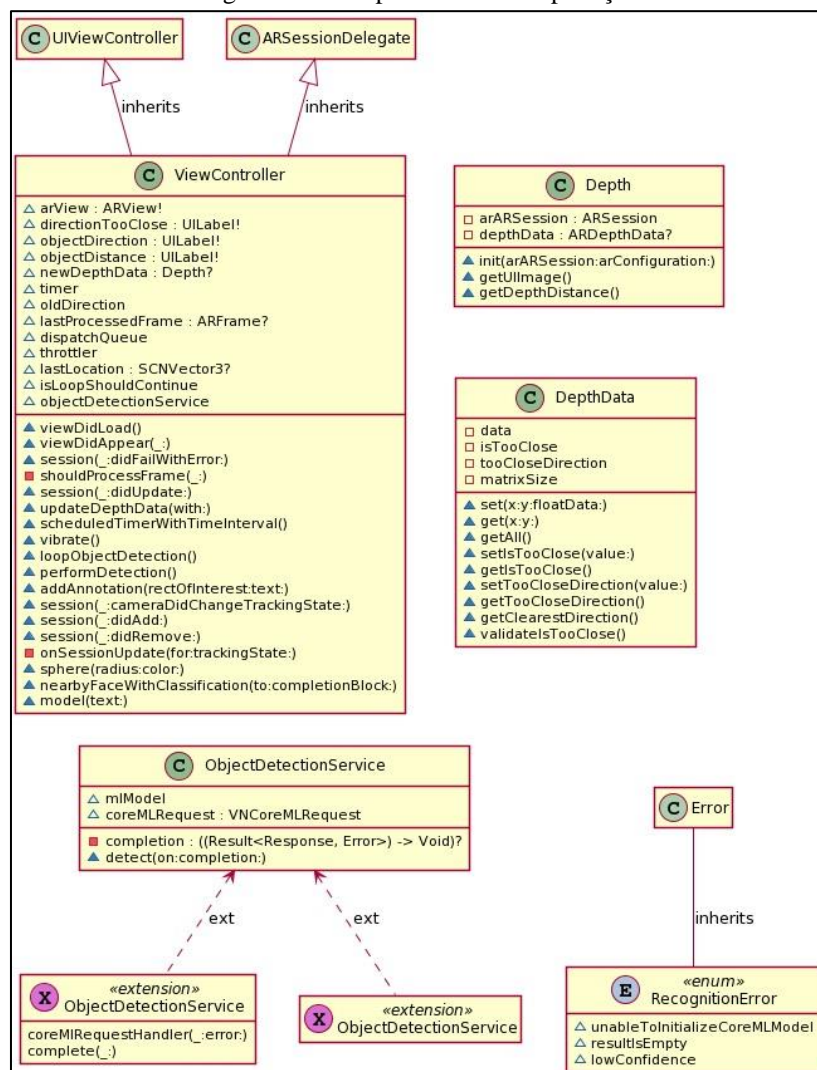
Fonte: elaborado pelo autor.

Para que o usuário não colida com algum obstáculo, o celular emite vibrações diferentes com o intuito de avisar para qual lado o usuário deve ir. Caso a direção com obstáculo esteja na esquerda, o celular emite uma vibração mais fraca, indicando que a direção livre pode estar na direita. Caso a direção com obstáculo esteja na direita, o celular emite uma vibração mais forte, indicando que a direção livre pode estar na esquerda.

A detecção de objetos acontece a cada frame da câmera, e utiliza apenas o reconhecimento por imagens. Quando objeto é detectado, uma esfera e um texto em 3D com o nome do objeto são adicionados ao ambiente virtual construído pelo aplicativo. Com a referência do objeto localizado no mundo virtual, essa âncora passa a emitir o som de uma voz falando o nome do objeto. Esse som apresenta noções de proximidade com o objeto, ou seja, quando o usuário se aproxima ou se distancia do objeto, o som fica mais próximo ou mais longe respectivamente.

A implementação foi separada em algumas classes, divididas entre o gerenciamento das informações obtidas pelo sensor LiDAR sendo estas as classes `Depth` e `DepthData`, e o serviço responsável pela detecção de objetos como a classe `ObjectDetectionService`. Todo o código é chamado pela classe principal `ViewController`, responsável por gerenciar a chamada de cada função e elementos de interface e câmera. A Figura 9 demonstra as principais classes citadas.

Figura 9 - Principais classes da aplicação



Fonte: elaborado pelo autor.

As atualizações de informações de distância obtidas pelo sensor LiDAR são atualizadas pela função `session(_didUpdate)`, que é chamada automaticamente quando há uma mudança de frame. A função então chama outra função denominada `updateDepthData`, que busca cada informação necessária e valida se há colisões. A função de sessão também é responsável por validar se o dispositivo está sendo movimentado na velocidade necessária, alterando a variável `isShouldLoopContinue` caso seja necessário. Essa variável mantém ou aborta o loop da função `performDetection`, responsável pela detecção de objetos.

3.2 EVITANDO COLISÕES

Utilizando as informações de distâncias captadas pelo LiDAR, é possível saber quando o usuário está próximo de colidir em algum obstáculo. Para acessar essas informações, o ARKit fornece a o parâmetro *sceneDepth*, configurável na inicialização do aplicativo. Com essa configuração, o ARKit traz uma matriz de dimensões 192x256, onde cada um dos pontos possui uma informação de distância. A Figura 10 demonstra uma simulação reduzida da matriz gerada pelo LiDAR, onde o celular se encontra a uma distância de 40cm da parede e à 25cm de uma caixa.

Figura 10 - Simulação da matriz gerada pelo LiDAR



0.40	0.40	0.39	0.38	0.40	0.38	0.39	0.38
0.40	0.39	0.40	0.38	0.38	0.39	0.39	0.39
0.39	0.38	0.38	0.39	0.39	0.38	0.38	0.38
0.39	0.38	0.38	0.39	0.38	0.37	0.38	0.38
0.40	0.39	0.39	0.40	0.39	0.38	0.39	0.38
0.39	0.39	0.39	0.38	0.39	0.38	0.38	0.38
0.27	0.27	0.28	0.27	0.28	0.28	0.27	0.27
0.26	0.25	0.25	0.25	0.26	0.25	0.25	0.26
0.26	0.25	0.25	0.25	0.25	0.25	0.25	0.26
0.26	0.25	0.25	0.25	0.26	0.25	0.25	0.26
0.26	0.25	0.25	0.25	0.26	0.25	0.25	0.26
0.26	0.25	0.25	0.25	0.26	0.25	0.25	0.26
0.24	0.23	0.24	0.23	0.22	0.22	0.22	0.23
0.18	0.19	0.19	0.20	0.20	0.20	0.21	0.21

Fonte: elaborado pelo autor.

Informações geradas pelo LiDAR são tratadas pela classe *Depth*, que percorre o mapa de profundidade do ARKit detectando colisões e preenchendo as informações de distância na matriz, retornando um objeto da classe *DepthData*, que armazena todas as informações de profundidade do *frame* em questão. O Quadro 4 mostra a busca dos valores do LiDAR através da classe *Depth*.

Quadro 4 - Busca de informações do LiDAR

```

if let newDepthData = newDepthData {
    // Busca dados de profundidade do LiDAR
    let data = newDepthData.getDepthDistance()
    // Busca informação de possível colisão com alguma direção
    let isTooClose = data.getIsTooClose()
    // Busca direção da colisão caso exista
    let direction = data.getTooCloseDirection()

    // Troca informação da direção com colisão
    if (direction != oldDirection){
        oldDirection = direction
    }

    // Valida se há colisão e se timer não está sendo executado
    if (isTooClose && !timer.isValid){
        // Chama função para iniciar timer
        scheduledTimerWithTimeInterval()
    }
    // Caso não exista mais colisão, para execução do timer
    else if (!isTooClose && timer.isValid) {
        timer.invalidate()
    }
}

```

Fonte: elaborado pelo autor.

Para orientar o usuário sobre a melhor direção a seguir para evitar uma colisão, divide-se a matriz pela metade verticalmente, tendo-se assim 96 pontos em cada lado. O lado cujo a média dos valores for menor, será o indicado pelo aplicativo para que o usuário desvie. Para que o aplicativo indique uma direção, algum dos pontos da matriz deverá ser menor ou igual a 50cm.

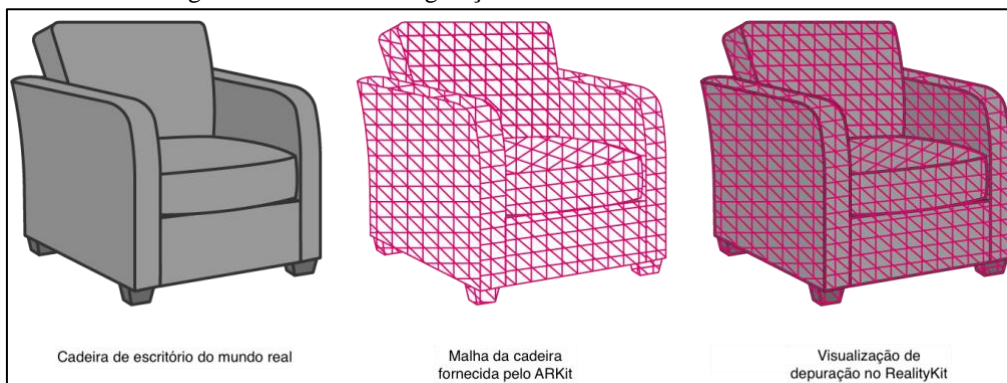
Sabendo a direção para qual o usuário deve desviar, resta enviar um retorno ao usuário a indicando. Para isso, utiliza-se duas diferentes formas de vibração disponíveis em dispositivos iOS. A primeira, indicando que o usuário deve virar à direita, utiliza um motor vibratório presente nos iPhones, chamado de Taptic Engine. Existem diferentes vibrações possíveis utilizando esse motor vibratório, mas a maioria está relacionada apenas a interação geral do usuário nos aplicativos (curtir uma foto em uma rede social ou enviar uma mensagem em algum aplicativo por exemplo). Como não há uma vibração específica relacionada a direções, utilizou-se uma vibração que indica colisão.

E por fim, para orientar que o usuário deve virar à esquerda, uma vibração simples acontece no dispositivo, não utilizando a resposta tátil como no caso anterior. Utilizar métodos de vibração diferentes garante que o usuário saberá identificar a diferença entre as vibrações, diminuindo assim as chances de que este vá para a direção incorreta. As vibrações acontecem a cada 1 segundo enquanto a informação da colisão existir, acionadas por uma função de timer.

3.3 GERANDO UMA MALHA NO AMBIENTE DETECTADO

O ARKit permite que, através dos dados obtidos pelo LiDAR, gere-se uma malha no ambiente detectado pelo sensor e que seja possível a exibir na tela utilizando o RealityKit. Como o LiDAR obtém rapidamente informações de profundidade de uma grande área em frente ao usuário, o ARKit pode estimar o formato do mundo real sem que haja necessidade do usuário se mover (APPLE, 2021h). A Figura 11 demonstra como o RealityKit levanta informações do mundo real através do ARKit e cria uma visualização de depuração.

Figura 11 - Processo de geração da malha no ambiente detectado



Fonte: Apple, (2021h).

Cada três vértices na malha formam um triângulo, chamado de face. Cada face possui informações designadas pelo ARKit, como classificação do objeto e suas coordenadas. Na aplicação são utilizadas as informações de distância da face com relação ao dispositivo do usuário, apenas para depuração exibidas em um quadro no canto superior da tela. A busca pelas informações de distância ocorre de forma assíncrona, já que a rotina envolve um processamento exaustivo.

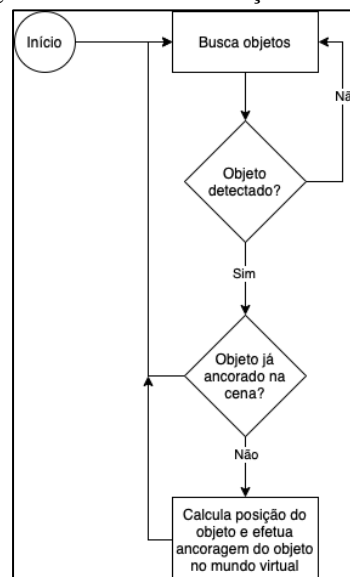
3.4 DETECTANDO OBJETOS

A classificação de objetos fornecida pela malha gerada possui uma limitação nas quantidades de objetos reconhecida, contando apenas com teto, chão, portas, assentos, mesas e paredes. Como alternativa, utilizou-se a detecção de objetos através de um modelo de aprendizado de máquina utilizando imagens. O modelo utilizado foi o YOLOv3, fornecido no site da Apple, treinado para localizar e classificar 80 tipos diferentes de objetos presentes no frame da câmera.

O modelo já está pronto para ser utilizado com o *framework* Vision, e ao detectar o objeto, fornece um vetor contendo índices de confiança e coordenadas normalizadas para gerar uma *bounding box* ao redor do objeto em questão. A função que processa a detecção de objetos é executada em loop, sendo chamada no máximo uma vez por segundo durante toda a execução.

Cada *frame* capturado pela câmera é enviado para a função de detecção, que realiza a busca por objetos reconhecíveis. Com o objetivo de evitar erros na detecção, define-se um limite de velocidade de movimentação do dispositivo. Caso o usuário esteja movendo o celular mais rápido do que 0.0085 frames por coordenada, a detecção não é realizada. Para que haja a classificação de um objeto, é necessário que este possua no mínimo 80% de confiança no resultado apresentado pelo modelo. A Figura 12 demonstra o fluxo de detecção dos objetos.

Figura 12 - Fluxo de detecção de objetos

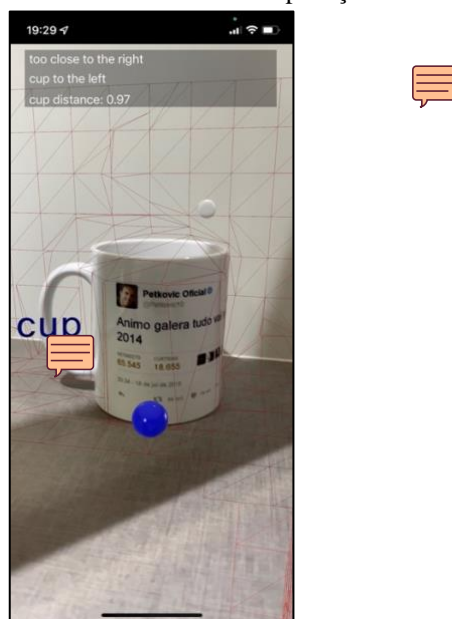


Fonte: elaborado pelo autor.

O último passo após a detecção é dizer ao usuário qual foi o objeto detectado e orientá-lo a respeito de sua direção. Para isso, utiliza-se as coordenadas *bounding box* gerada pelo modelo para destacar a área de interesse a qual o objeto está inserido. Com as coordenadas, é realizado um *raycast* contra a malha gerada pelo ARKit, e utiliza-se o retorno do ponto mais próximo da câmera. Esse processo converte as coordenadas da câmera para o ambiente virtual gerado pelo ARKit, e torna mais fácil adicionar uma âncora no ambiente, que armazenará as informações do objeto detectado.

Com o ponto obtido, é realizada a busca pelo centro da face mais próxima ao ponto em questão. Com o retorno da face, é possível obter também sua distância com relação à câmera do celular, informação que é exibida em tela apenas para depuração. Após todas as informações de coordenadas serem obtidas, duas âncoras precisam ser adicionadas ao mundo virtual, sendo elas o elemento de texto contendo o nome do objeto detectado, e uma esfera que indicará o centro da face mais próxima do objeto detectado. A Figura 13 demonstra uma xícara detectada pela aplicação, com as âncoras de texto e esfera próximas a sua respectiva localização.

Figura 13 - Xícara detectada na aplicação



Fonte: elaborado pelo autor.

Tanto a âncora da esfera quanto a de texto possuem um tempo de expiração, que poderia ser utilizado por exemplo para “invalidar” uma detecção de certo objeto que se move, como um gato ou cachorro por exemplo, fazendo com que o aplicativo atualize a localização do objeto constantemente. Nesta implementação, o tempo padrão está definido para 10 segundos para todos os objetos, e após esse tempo a âncora é removida do mundo gerado.

3.5 OBTENDO DIREÇÕES DO OBJETO

Foram implementadas duas formas diferentes para orientar o usuário na direção do objeto detectado. A primeira forma utiliza as coordenadas da *bounding box* gerada pelo modelo, e busca se as coordenadas estão à esquerda ou direita da tela. Com esse método, as direções para o objeto só são exibidas caso este ainda esteja no campo de visão e sendo detectado pela aplicação. Essa implementação permanece ativa, mas é utilizada apenas nas informações demonstradas no canto superior da tela para fins de depuração.

Aproveitando o benefício da malha gerada pelo ARKit, e consequentemente das âncoras colocadas no mundo virtual, houve a possibilidade de utilizar o recurso de áudio espacial. O áudio espacial traz imersão para o usuário, de forma que seja possível estimar as direções na qual um áudio é reproduzido, bem como se este está longe ou perto. Desta forma, uma das âncoras ficará emitindo um som repetidamente, e o usuário saberá a sua direção e sua distância aproximada. O Quadro 5 demonstra a chamada da função para reproduzir o áudio espacial.

Quadro 5 - Função para reprodução de áudio espacial

```
do { let audioResource = try AudioFileResource.load(named: text + ".mp3",  
                                                    in: nil,  
                                                    inputMode: .spatial,  
                                                    loadingStrategy: .preload,  
                                                    shouldLoop: true)  
model.playAudio(audioResource)
```

Fonte: elaborado pelo autor.

Para identificar de qual objeto se trata, o áudio emitido pela âncora será o nome do objeto. Para isso, foram gerados arquivos de áudio utilizando uma ferramenta *text to speech*, e salvos no projeto para utilização. Ao adicionar a âncora, o nome do objeto detectado é passado por parâmetro, e a classe `AudioFileResource` do RealityKit se encarrega de carregar o arquivo de áudio e realizar a reprodução em áudio espacial.

4 RESULTADOS

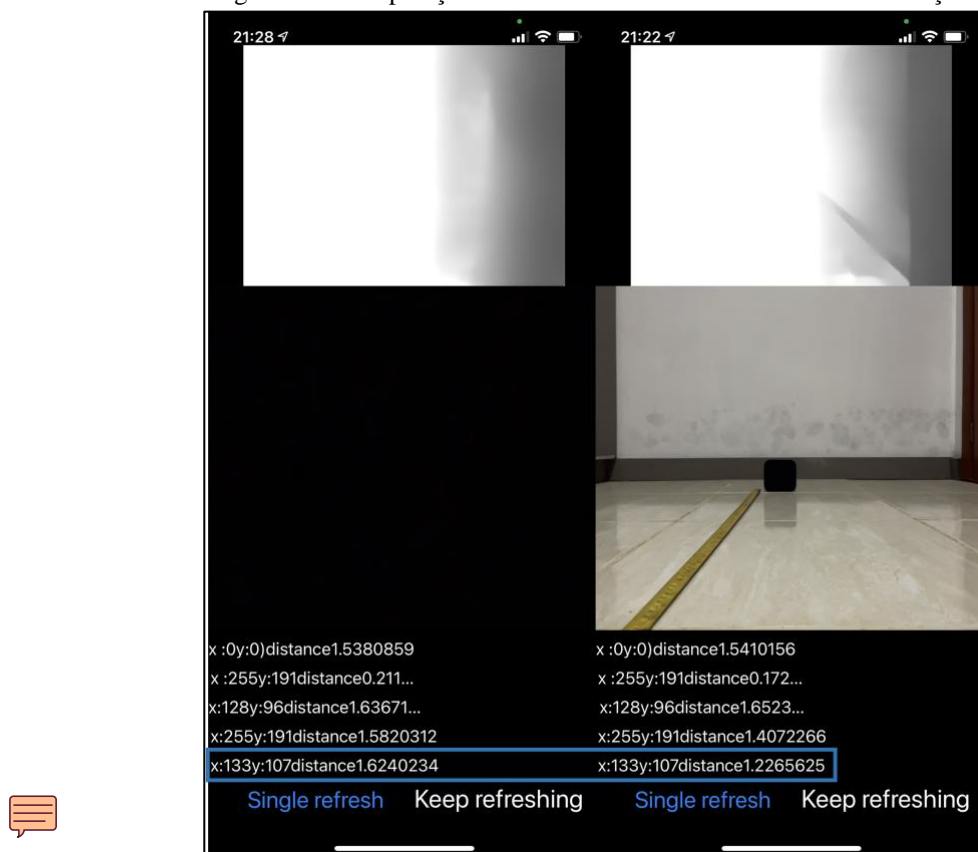
A seção de resultados foi dividida em quatro partes. Na primeira são apresentados testes do sensor LiDAR em ambientes com pouca iluminação. Na segunda são demonstrados testes de desempenho da aplicação, focando no consumo de processamento e bateria do dispositivo. A terceira apresenta testes realizados com a detecção de objetos e a precisão da sua localização e distância em alguns cenários. E por fim são citados cenários onde a aplicação detecta vários objetos ao mesmo tempo, bem como seu impacto na orientação do usuário para localizar o objeto.

4.1 TESTES COM O LIDAR

Alguns testes iniciais do desenvolvimento da aplicação se voltaram para a acurácia com relação às distâncias detectadas com o sensor LiDAR. Com o auxílio de uma trena, validou-se a distância entre o dispositivo e uma parede, e com diferentes objetos entre eles. O ponto mais importante nesse caso é saber se as informações de distância continuam as mesmas em um ambiente com baixa ou nenhuma iluminação, quando comparadas com um ambiente bem iluminado.

Para esse teste, utilizou-se uma aplicação teste, e colocou-se um objeto a 1,20m de distância do dispositivo. Dois cenários foram validados, sendo o primeiro com o ambiente totalmente sem iluminação, e o segundo com as luzes acesas. Com as informações obtidas, se pode perceber que houve uma variação de cerca de 40cm entre os dois cenários. A Figura 14 demonstra o teste realizado, com os ambientes sem iluminação e com iluminação respectivamente.

Figura 14 - Comparação de dados do LiDAR em diferentes iluminações



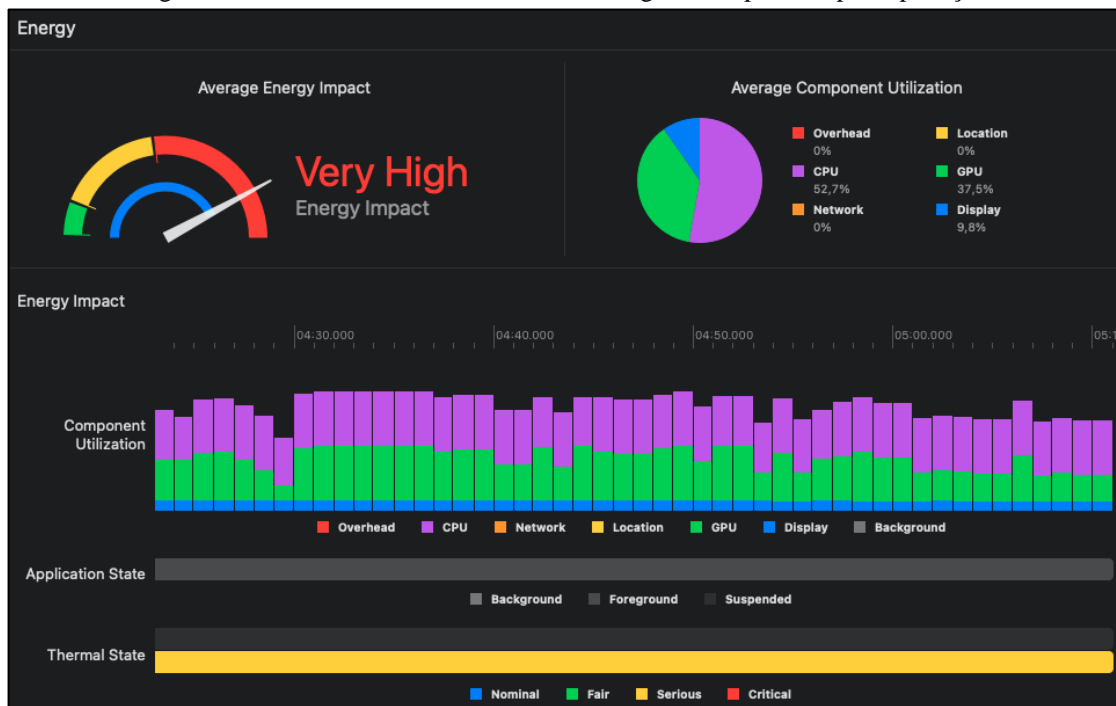
Fonte: elaborado pelo autor.

Mesmo com a variação de 40cm apresentada em ambientes sem iluminação, a detecção de colisões pode não ser afetada, visto que essa situação ocorreu ao detectar obstáculos a uma distância consideravelmente longa. Com a aplicação programada para alertar o usuário em caso de obstáculos mais próximos do que 50cm, o problema em questão não deverá ocorrer. Outro teste possível para essa situação é a utilização de um equipamento ou sensores do próprio aparelho para medir a intensidade de iluminação do ambiente, validando quais limites de pouca ou muita luz permitem que aplicação continue estimando as distâncias de forma correta.

4.2 TESTES DE DESEMPENHO

Com o uso da aplicação destinado a um smartphone, desempenho e eficiência energética são pontos que requerem uma maior atenção. Ao longo do uso da aplicação, é possível sentir uma elevação da temperatura do aparelho, bem como perceber uma queda considerável da bateria. Considerando a quantidade de processamento necessária para que a aplicação execute tarefas que envolvam a tratativa de dados do LiDAR e a detecção de objetos, pode ser algo relativamente compreensível, mas que também pode acarretar problemas para o usuário final. A Figura 15 demonstra um gráfico gerado pelo ambiente de desenvolvimento Xcode, utilizado para a construção da aplicação, demonstrando o uso de energia do aplicativo.

Figura 15 - Gráficos demonstrando uso de energia do dispositivo pela aplicação



Fonte: elaborado pelo autor.

A Figura 15 demonstra que boa parte do impacto gerado na energia do dispositivo vem do uso de processador, seguido pelo uso da unidade de processamento gráfico. De acordo com as legendas, também se observa que a temperatura atinge um estado considerado sério, confirmando que o uso por cerca de 5 minutos já é suficiente para elevar a temperatura do dispositivo. Nos testes realizados, segurar o aparelho com essa temperatura elevada, mesmo que por pouco tempo, já causa um desconforto na mão do usuário, o que pode ser considerado um problema visto que esta é uma das suas principais formas de uso. Além da consequência direta do alto consumo de energia em descarregar mais rapidamente a bateria do aparelho, limitando assim a sua mobilidade.

4.3 TESTES DE DETECÇÃO DE OBJETOS



A detecção de objetos ocorre de forma satisfatória quando o dispositivo está próximo dos objetos a serem localizados. Em distâncias maiores do que 50cm e com objetos relativamente pequenos como uma xícara, algumas imprecisões podem ser percebidas, sendo principalmente o caso do posicionamento dos objetos detectados. Para cada objeto detectado é gerada uma *bounding box*, e através desta também é gerada a âncora de localização do objeto no mundo detectado. Como a âncora é sempre inserida em uma face que esteja mais próxima da *bounding box* na malha gerada pelo ARKit, caso o ponto calculado para sua inserção fique fora do objeto, a distância com relação ao usuário pode ficar imprecisa. A Figura 16 demonstra um exemplo dessa situação, onde o dispositivo estava a 50cm da xícara.

Figura 16 - Imprecisão na distância do objeto



Fonte: elaborado pelo autor.

Mesmo com a âncora em uma posição não muito próxima do objeto, ainda é possível saber quais orientações seguir para acessá-lo. Infelizmente, a imprecisão pode acarretar uma necessidade do usuário “buscar” onde o objeto realmente está com as próprias mãos, diminuindo a confiança na aplicação. Em contrapartida, como a aplicação refaz a detecção e ancoragem da localização do objeto a cada 10 segundos, é possível que ao se aproximar do objeto o usuário possua uma localização mais precisa.

4.4 TESTES DE DETECÇÃO EM AMBIENTES COM VÁRIOS OBJETOS


Caso o usuário esteja em um ambiente com muitos objetos, pode ser difícil entender e distinguir a direção de som de cada um deles. Em um cenário onde todos os objetos estão em uma mesa e próximos um do outro por exemplo, cada objeto detectado ganhará uma âncora e emitirá um som com seu respectivo nome. A não ser que o usuário aproxime muito do dispositivo do objeto alvo, evitando assim que outros apareçam na visão da câmera e possam ser detectados pela aplicação, o áudio se tornará relativamente poluído, tendo sua eficácia para o direcionamento do usuário consideravelmente reduzida.

O tempo de expiração de cada detecção pode ajudar também nessa situação, visto que ao se aproximar do objeto alvo o usuário vai deixando de capturar as imagens que demonstram outros objetos. De mesma forma, objetos iguais não possuem suas âncoras adicionadas mais de uma vez, reduzindo também as chances de tornar o ambiente e as orientações muito repetitivas ou poluídas.

5 CONCLUSÕES

Diante dos resultados apresentados é possível concluir que a aplicação atinge seus principais objetivos, de pesquisar formas de auxiliar a orientação de pessoas com algum tipo de deficiência visual. O LiDAR, até então restrito a carros que dirigem sozinhos ou a equipamentos profissionais, está sendo introduzido na vida das pessoas através de dispositivos móveis. Uma possível popularização desse sensor por outras fabricantes, bem como seu uso em aparelhos vestíveis como um par de óculos por exemplo, pode facilitar ainda mais o desenvolvimento de aplicações desse tipo, e possibilitar cada vez mais a inclusão em nossa sociedade.

Em relação ao desenvolvimento da aplicação, grande parte da implementação é facilitada pelos *frameworks* disponibilizados de forma nativa pela Apple. Com algumas configurações na inicialização da aplicação e com a chamada de alguns métodos, tem-se acesso a informações do sensor LiDAR como suas distâncias e mapas de profundidade, bem como reconhecimento do ambiente no mundo real e suas formas físicas, e reconhecimento de objetos utilizando um

 modelo de aprendizagem de máquina já treinado. Realidade aumentada é um dos temas que recebe boa atenção pela Apple, visto que há uma riqueza de funcionalidades nesse sentido encontradas tanto nas documentações da empresa para desenvolvedores, quanto na divulgação de aplicações para o usuário final que utilizem os sensores disponíveis nos seus dispositivos.

Embora haja caminho para as mais diversas melhorias na aplicação desenvolvida, como melhora na performance, precisão na detecção de objetos e utilização do LiDAR nesse processo, a aplicação demonstra que se torna cada vez mais simples integrar a tecnologia para ajudar quem necessita. Para extensões desse trabalho propõe-se:

- a) implementar validação de diferentes planos, como escadas;
- b) melhorar a precisão da distância de um objeto detectado;
- c) utilizar também o LiDAR na detecção de objetos;
- d) utilizar alternativas para evitar que muitos objetos detectados dificultem a orientação do usuário;
- e) integrar a aplicação com outro dispositivo vestível, como um *smartwatch*, tornando mais nítido o retorno de dados e direções para o usuário;
- f) invalidar âncoras de objetos detectados em diferentes tempos de acordo com o grupo de cada objeto, como objetos estáticos ou móveis por exemplo;
- g) gerar os áudios com o nome de cada objeto reconhecível pela aplicação de forma dinâmica na própria aplicação;
- h) utilizar um equipamento ou sensores do próprio aparelho para medir a intensidade de iluminação do ambiente para validar quais limites de luminosidade permitem que a aplicação continue detectando distâncias corretamente;
- i) ampliar os testes a respeito do consumo de bateria pela aplicação, comparando com o uso normal do dispositivo pelo usuário.

REFERÊNCIAS

APPLE. **AnchorEntity**, 2021a. Disponível em: <https://developer.apple.com/documentation/realitykit/anchorentity>. Acesso em: 21 nov. 2021.

APPLE. **ARKit**, 2021b. Disponível em: <https://developer.apple.com/documentation/arkit>. Acesso em: 11 abr. 2021.

APPLE. **Core ML**, 2021c. Disponível em: <https://developer.apple.com/documentation/coreml>. Acesso em: 22 nov. 2021.

APPLE. **Introducing ARKit 4**, 2021d. Disponível em: <https://developer.apple.com/augmented-reality/arkit/>. Acesso em: 11 abr. 2021.

APPLE. **iPhone 12 Pro e iPhone 12 Pro Max**, 2021e. Disponível em: <https://www.apple.com/br/iphone-12-pro/>. Acesso em: 21 mar. 2021.

APPLE. **RealityKit**, 2021f. Disponível em: <https://developer.apple.com/documentation/realitykit/>. Acesso em: 10 nov. 2021.

APPLE. **RealityKit 2 Overview**, 2021g. Disponível em: <https://developer.apple.com/augmented-reality/realitykit/>. Acesso em: 10 nov. 2021.

APPLE. **Visualizing and Interacting with a Reconstructed Scene**, 2021h. Disponível em: https://developer.apple.com/documentation/arkit/content_anchors/visualizing_and_interacting_with_a_reconstructed_scene. Acesso em: 11 abr. 2021.

BAI, J. *et al.* Wearable Travel Aid for Environment Perception and Navigation of Visually Impaired People. **Electronics**, [S.L.], v. 8, n. 6, p. 697, 20 jun. 2019. MDPI AG. Disponível em: <http://dx.doi.org/10.3390/electronics8060697>. Acesso em: 13 mar. 2021.

BOURNE, R. Causes of blindness and vision impairment in 2020 and trends over 30 years, and prevalence of avoidable blindness in relation to VISION 2020: the Right to Sight: an analysis for the Global Burden of Disease Study. **Lancet Global Health**, Anglia Ruskin University, v. 9, n. 2, p. 144-160, fev./2021. Disponível em: [https://www.thelancet.com/journals/langlo/article/PIIS2214-109X\(20\)30489-7/fulltext](https://www.thelancet.com/journals/langlo/article/PIIS2214-109X(20)30489-7/fulltext). Acesso em: 21 mar. 2021.

BRUNES, A.; HANSEN, M.; HEIR, T. Loneliness among adults with visual impairment: prevalence, associated factors, and relationship to life satisfaction. **BMC Public Health**, Health Qual Life Outcomes, v.19, n. 17, fev. 2019. Disponível em: <https://hql.o.biomedcentral.com/articles/10.1186/s12955-019-1096-y>. Acesso em: 21 mar. 2021.

EUROSENSORS CONFERENCE, 32., 2018, Graz. Review of LiDAR Sensor Data Acquisition and Compression for Automotive Applications. Graz: Mdpi, 2018. 4 p. Disponível em: <https://doi.org/10.3390/proceedings2130852>. Acesso em: 27 mar. 2021.

- EVERYPOINT. EveryPoint Gets Hands-On with Apple's New Lidar Sensor. **EveryPoint**, 2020. Disponível em: <https://everypoint.medium.com/everypoint-gets-hands-on-with-apples-new-lidar-sensor-44eeb38db579>. Acesso em: 12 abr. 2021.
- GREGERSEN, E. Lidar. **Encyclopedia Britannica**, 2016. Disponível em: <https://www.britannica.com/technology/lidar>. Acesso em: 11 abr. 2021.
- JIANG, Rui; LIN, Qian; QU, Shuhui. **Let Blind People See: Real-Time Visual Recognition with Results Converted to 3D Audio**. Palo Alto: Stanford University, 2016. 7 p. Disponível em: http://cs231n.stanford.edu/reports/2016/pdfs/218_Report.pdf. Acesso em: 13 mar. 2021.
- LARSSON, Niklas; RUNESSON, Hampus. **A study on the use of ARKit to extract and geo-reference floor plans**. 2021. 59 f. Tese (Mestrado) - Curso de Tecnologia da Informação, Department Of Computer And Information Science, Linköpings Universitet, Linköping, 2021. Disponível em: <http://liu.diva-portal.org/smash/get/diva2:1575514/FULLTEXT01.pdf>. Acesso em: 10 nov. 2021.
- MUHADI, N. A. *et al.* The Use of LiDAR-Derived DEM in Flood Applications: a review. **Remote Sensing**, [S.L.], v. 12, n. 14, p. 2308, 18 jul. 2020. MDPI AG. Disponível em: <http://dx.doi.org/10.3390/rs12142308>. Acesso em: 15 maio 2021.
- NOWACKI, P.; WODA, M.; Capabilities of ARCore and ARKit Platforms for AR/VR Applications. **Advances In Intelligent Systems And Computing**, [S.L.], p. 358-370, 12 maio 2019. Springer International Publishing. http://dx.doi.org/10.1007/978-3-030-19501-4_36.
- PETIT, F. THE BEGINNINGS OF LIDAR – A TIME TRAVEL BACK IN HISTORY. **Blickfeld**, 2020. Disponível em: <https://www.blickfeld.com/blog/the-beginnings-of-lidar/>. Acesso em: 11 abr. 2021.
- PHAM, H.; LE, T.; VUILLERME, N. Real-Time Obstacle Detection System in Indoor Environment for the Visually Impaired Using Microsoft Kinect Sensor. **Journal Of Sensors**, [S.L.], v. 2016, p. 1-13, 2016. Hindawi Limited. Disponível em: <http://dx.doi.org/10.1155/2016/3754918>. Acesso em: 13 mar. 2021.
- PLIKYNAS, D. *et al.* Indoor Navigation Systems for Visually Impaired Persons: mapping the features of existing technologies to user needs. **Sensors**, [S.L.], v. 20, n. 3, p. 636, 23 jan. 2020. MDPI AG. Disponível em: <http://dx.doi.org/10.3390/s20030636>. Acesso em: 21 mar. 2021.
- RENAUD, J.; BÉDARD, E. Depression in the elderly with visual impairment and its association with quality of life. **Clinical Interventions In Aging**, [S.L.], p. 931, jul. 2013. Informa UK Limited. Disponível em: <http://dx.doi.org/10.2147/cia.s27717>. Acesso em: 21 mar. 2021.
- USAU, V. Core Facts about LiDAR You Should Know. **Emerline**, 2020. Disponível em: <https://emerline.com/blog/core-facts-about-lidar-you-should-know>. Acesso em: 11 abr. 2021.
- WORLD HEALTH ORGANIZATION. Blindness and vision impairment. **World Health Organization**, 2021. Disponível em: <https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment>. Acesso em: 21 mar. 2021.
- WORLD HEALTH ORGANIZATION. ICD-11 FOR MORTALITY AND MORBIDITY STATISTICS (VERSION : 09/2020). **IDC-11**, 2020. Disponível em: <https://icd.who.int/browse11/l-m/en#/http%3a%2f%2fid.who.int%2fid%2fentity%2f1103667651>. Acesso em: 21 mar. 2021.
- WU, J. An automatic procedure for vehicle tracking with a roadside LiDAR sensor. **ITE Journal**, [S.L.], v. 88, n. 11, p. 32-37, nov. 2018. Disponível em: <https://www.westernite.org/awards/vanwagoner/2019%20-%20Van%20Wagoner.pdf>. Acesso em: 21 mar. 2021.