

APLICATIVO MÓVEL QUE AUXÍLIA PESSOAS COM DEFICIÊNCIA VISUAL NO DESLOCAMENTO DE AMBIENTES INTERNOS

Guilherme Barth, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil
guibarth@furb.br, dalton@furb.br

Resumo: Este artigo apresenta um aplicativo iOS que utiliza o sensor LiDAR e o ML Kit da Google para fazer a identificação dos objetos de um ambiente interno e orientar o usuário de onde eles estão localizados. Foi utilizado um iPhone 13 Pro como dispositivo móvel. O projeto foi desenvolvido em Swift, utilizando o XCode v14.5 para suportar as últimas bibliotecas do LiDAR disponibilizadas pela própria Apple. Os testes mostraram que é possível fazer essa identificação de objetos com algumas limitações onde o LiDAR não consegue capturar a existência de um objeto por conta do seu formato não padrão. Os comandos de voz do aplicativo conseguem orientar adequadamente o usuário, porém o volume ficou abaixo do esperado. Além disso o usuário deve se comunicar na língua inglesa.

Palavras-chave: Sensor LiDAR. ML Kit. Deficiência Visual.

1 INTRODUÇÃO

Nos tempos atuais, diversos estudos e implementações vem sendo desenvolvidas a partir de uma tecnologia bastante conhecida: o laser. Um exemplo é o Light Detection and Ranging (LiDAR), que está se tornando cada vez mais popular e utilizado em diversos estudos, principalmente na área da topografia, como por exemplo, em planejamentos costeiros, florestas, urbanos e de automóveis (PEREIRA, 2022). O LiDAR é um dos termos utilizados para designar essa nova tecnologia de sensoriamento remoto, que também pode ser encontrada como Sistema de Varredura a Laser ou Perfilamento a Laser (PEREIRA, 2022). “O princípio de funcionamento do sistema de varredura laser consiste na emissão de um pulso laser de uma plataforma (aérea, terrestre ou orbital) com uma elevada frequência de repetição. O tempo de retorno dos pulsos laser entre a plataforma e os alvos é medido pelo sensor, permitindo a estimativa destas distâncias.” (PEREIRA, 2022, p. 232).

Para auxiliar a utilização do LiDAR, utiliza-se Machine Learnings para auxiliar na identificação dos objetos do mundo real (Apple, 2022b). Dentre elas, existe o Machine Learning Kit ou então “Kit de ML que é um SDK para dispositivos móveis que leva a experiência em aprendizado de máquina no dispositivo do Google a apps Android e iOS” (GOOGLE, 2022, p. 1). O Kit de ML consegue detectar até cinco objetos ao mesmo tempo em uma categoria de 400 objetos mais encontrado nas fotos (GOOGLE, 2022).

Freitas (2018) cita que a comunicação é essencial para a pessoa deficiente conseguir perceber a presença de outras pessoas, intenções de ajuda e orientação. Visando a parte técnica, existem algumas Application Programming Interface (APIs) que permitem realizar comandos vocais utilizando o Text to Speech (TTS), Speech to Text (STT) e Automatic Speech Recognition (ASR) (GOULART, 2016).

Conforme Silva (2019, p.1) “Para as pessoas sem deficiência, a tecnologia torna as coisas mais fáceis. Para as pessoas com deficiência, a tecnologia torna as coisas possíveis.”. Ainda de acordo com Silva (2019), o principal objetivo da tecnologia assistiva é proporcionar às pessoas com deficiência visual uma independência, qualidade de vida e inclusão social, por meio de melhorias na comunicação, mobilidade e autonomia no seu ambiente. Segundo Cambraia e Nazima (2021) a deficiência visual é um grave problema de saúde pública, visto que o índice de pessoas com deficiência visual aumenta conforme a população aumenta e envelhece, além de diminuir a qualidade de vida, existem dificuldades emocionais para se adaptarem e aumenta o risco de quedas e até mortes.

Pessoas com baixa visão ou cegas, precisam lançar mão de algum recurso que possibilite uma locomoção segura, sendo a bengala a opção mais conhecida e utilizada para a mobilidade, na medida em que serve para identificar obstáculos que se encontram à frente ou ao nível do chão; os chamados obstáculos rasteiros. Ocorre que no uso da bengala, por meio da técnica conhecida como varredura, o usuário a movimentada de um lado para o outro desenhando um arco à sua frente, mas precisa tocar no obstáculo para identificá-lo e somente depois desviar dele (CASTRO, 2019, p.12).

Diante disso, este trabalho apresenta um aplicativo de reconhecimento de estruturas utilizando o sensor LiDAR como protagonista do reconhecimento de superfícies, e a utilização do TTS e STT para realizar a comunicação com o usuário. Sendo o objetivo principal disponibilizar um aplicativo na plataforma iOS que auxilie pessoas com deficiência

visual a identificar objetos durante o seu deslocamento utilizando o LiDAR e o ML Kit. O objetivo específico é: avaliar se a medida de distância em que o objeto se encontra do dispositivo móvel é coerente com a realidade, e fazer a interação do usuário com o aplicativo por comandos de voz.



2 FUNDAMENTAÇÃO TEÓRICA

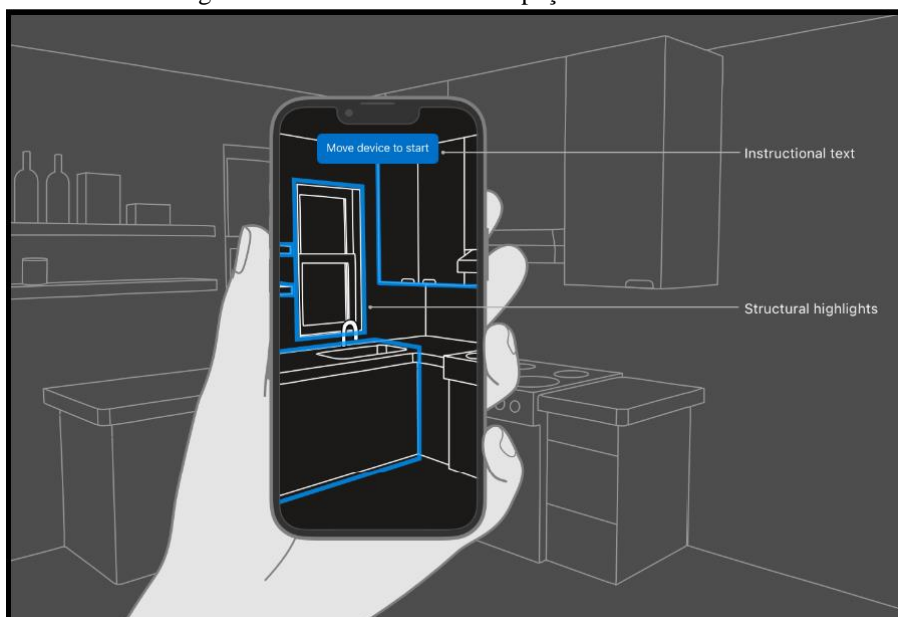
Nesta seção será apresentado os conceitos para o entendimento do projeto realizado. A subseção 2.1 descreve as principais características da biblioteca utilizada pelo LiDAR (Room Plan). Já a 2.2 aborda o ML Kit. Na subseção 2.3 é realizado um breve resumo sobre o reconhecimento de voz. Por fim, na subseção 2.4 são listados os trabalhos correlatos.

2.1 ROOM PLAN

De acordo com Ippolita (2022), ao utilizar dados de sensoriamento e fotogrametria o LiDAR produz dados de altíssima qualidade. Segundo a Apple (2022c), o Room Plan é um *framework* principalmente utilizado para criar modelos 3D do interior de uma sala usufruindo dos sensores dos dispositivos móveis, além de utilizar de Machine Learnings treinadas e os recursos de renderização do RealityKit para capturar o ambiente físico de uma sala interna. O RealityKit é um *framework* que simula e renderiza conteúdo 3D para os seus aplicativos de realidade aumentada (APPLE, 2022b). O Room Plan consegue inspecionar a câmera e aplicando o sensor LiDAR é possível fazer identificação como paredes, portas, janelas e objetos no geral. Conforme é percorrido o interior do local do usuário é possível visualizar o resultado da sala interna em uma pequena escala 3D (APPLE, 2022c). Na Figura 1 é mostrado o resultado da inspeção do Room Plan e a Figura 2 mostra como o RealityKit consegue adicionar um objeto no mundo virtual.

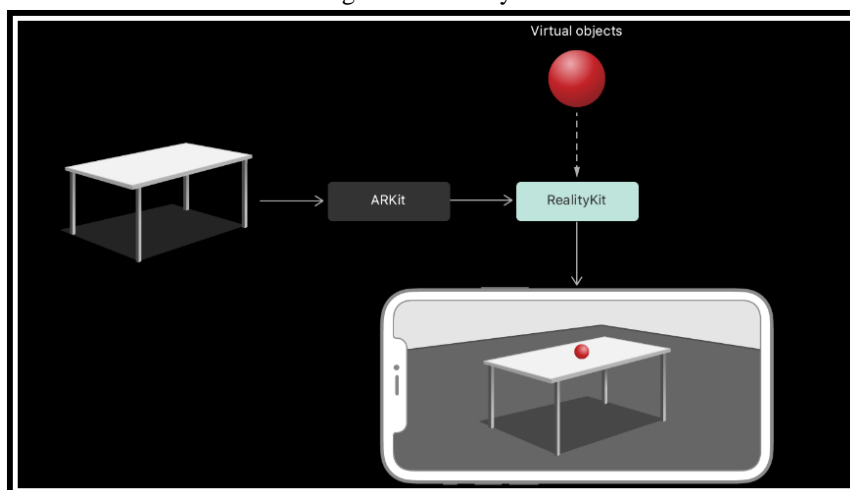
Ele possui diversos eventos que facilitam a sua compreensão e mantém a diferenciação de funcionalidades bem definida. Segundo a Apple (2022c) O `captureSession` possui diversas separações da sua funcionalidade, dentre elas temos o `didAdd` que é chamado toda vez que um objeto novo é encontrado, retornando todos os objetos já encontrados e o recém detectado. Possui também o `captureSession` com `didEndWith` que retorna o resultado do escaneamento, além de diversos outros delegados, controles de sessão e *layouts* que não foram utilizados nesse projeto.

Figura 1 – Resultado de uma inspeção do Room Plan



Fonte: Apple (2022c).

Figura 2 – RealityKit



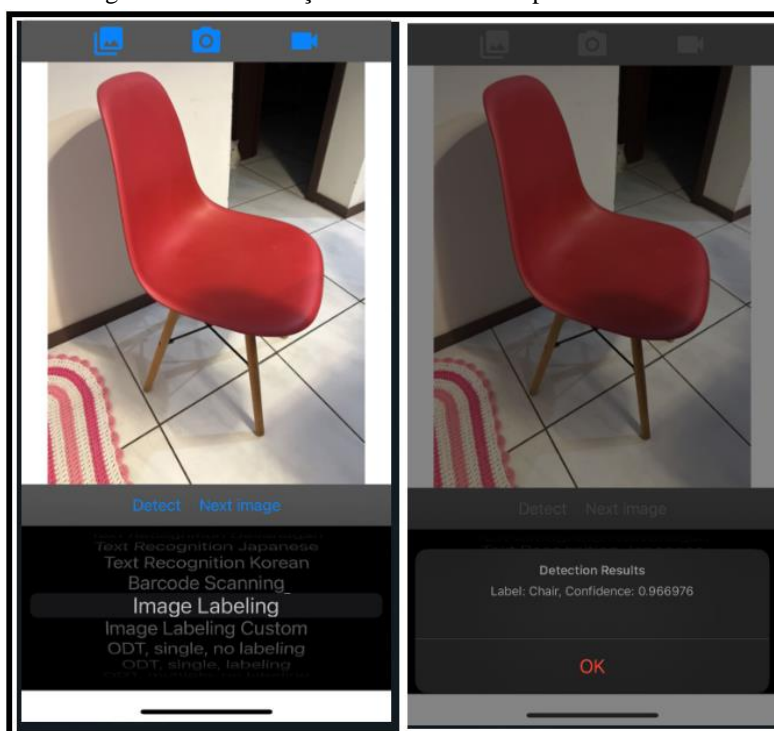
Fonte: Apple (2022b).

2.2 ML KIT

Segundo a Google (2022), o Kit de ML é um SDK que permite o usuário utilizar várias ferramentas de maneira muito prática por meio de um dispositivo móvel. Também é conhecido por ser um facilitador de resolução de desafios comuns ou conhecidos por ser um criador de experiências para os usuários. Dentre as suas funcionalidades o Kit de ML possui reconhecimento de texto, detecção facial, detecção de malha de rosto, detecção e rastreamento de objetos, reconhecimento de tinta digital, entre outros que estão recebendo atualizações (GOOGLE, 2022). Na Figura 3 é mostrado a apenas a identificação de objetos do ML Kit, porém ele pode ser explorado mais profundamente.

De acordo com a Google (2022), os modelos normais de reconhecimento de imagens utilizam um modelo poderoso de rotulagem de imagens de uso geral que permite reconhecer mais de 400 entidades que abrangem os conceitos mais encontrados nas fotos, isso com o modelo padrão o modelo utilizado na Figura 3. Caso seja necessário fazer o reconhecimento de conceitos mais específicos, a API aceita modelos personalizados pelo desenvolvedor que podem ser pré-treinados pelo TensorFlow Lite Model Maker ou por meio do TensorFlow. Esses modelos personalizados ficariam salvados e carregados juntamente com o aplicativo desenvolvido ou utilizar do Firebase Machine Learning conforme a necessidade. No caso desse projeto foi utilizado o modelo padrão.

Figura 3 – Identificação de uma cadeira a partir do ML Kit



Fonte: Digitalizado pelo autor.

2.3 SPEECH TO TEXT E TEXT TO SPEECH

Segundo Herchonvicz, Franco e Jasinski (2019), embora o sistema de reconhecimento de fala tenha sido aprimorado para utilização prática, alguns desafios permanecem. Por exemplo, o reconhecimento de fala em vários idiomas ainda não é resolvido devido à escassez de dados de conversação e textos com dicionários de pronúncia, a falta de convenções de linguagem e a lacuna entre a tecnologia e linguagem perícia. Em contrapartida, a Apple possui a *framework* AVFoundation que, dentre as várias funcionalidades, a principal utilizada nesse projeto é a que permite transformar textos em saídas de voz, garantindo assim a interação com o usuário final (APPLE, 2022a). Já o *framework* Speech permite fazer o processo contrário, ou seja, reconhecer palavras faladas ou em áudio gravado ao vivo (APPLE, 2022d). No Quadro 1 é apresentado um trecho do código do aplicativo que realiza a conversão de texto para fala utilizando o AVFoundation, sendo possível definir o idioma, a velocidade e o texto.

Durante a utilização do Speech encontrou-se uma dificuldade na sua utilização que pode ser observada no Quadro 9, onde o usuário fica limitado em certos aspectos de conversação e obrigando que o aplicativo trate as possíveis alterações da língua estrangeira.

Quadro 1 – Conversão de voz de texto para áudio

```
33
34 let synthesizer = AVSpeechSynthesizer()
35
36 override func viewDidLoad() {
37
38     let utterance = AVSpeechUtterance(string: "Para começar fale INICIAR BUSCA ou Start Scanning")
39     utterance.voice = AVSpeechSynthesisVoice(language: "pt-BR")
40     utterance.rate = 0.5
41
42     synthesizer.speak(utterance)
43
44     requestPermission()
45     startRecognition()
46
47 }
```

Fonte: Elaborado pelo autor.

2.4 TRABALHOS CORRELATOS

A seguir são apresentados três trabalhos correlatos. No Quadro 2 é apresentado o projeto de Rossi, Freitas e Reis (2019), que consiste em um mapeamento tridimensional em ambientes internos. No Quadro 3 é apresentado o trabalho de Franco (2018), que criaram um aplicativo para reconhecimento de objetos em imagens para pessoas portadoras de deficiência visual. Por fim, no Quadro 4 é descrito o trabalho de Pereira (2022), que compara mapas de suscetibilidade a deslizamentos com dados do LiDAR e VANT.

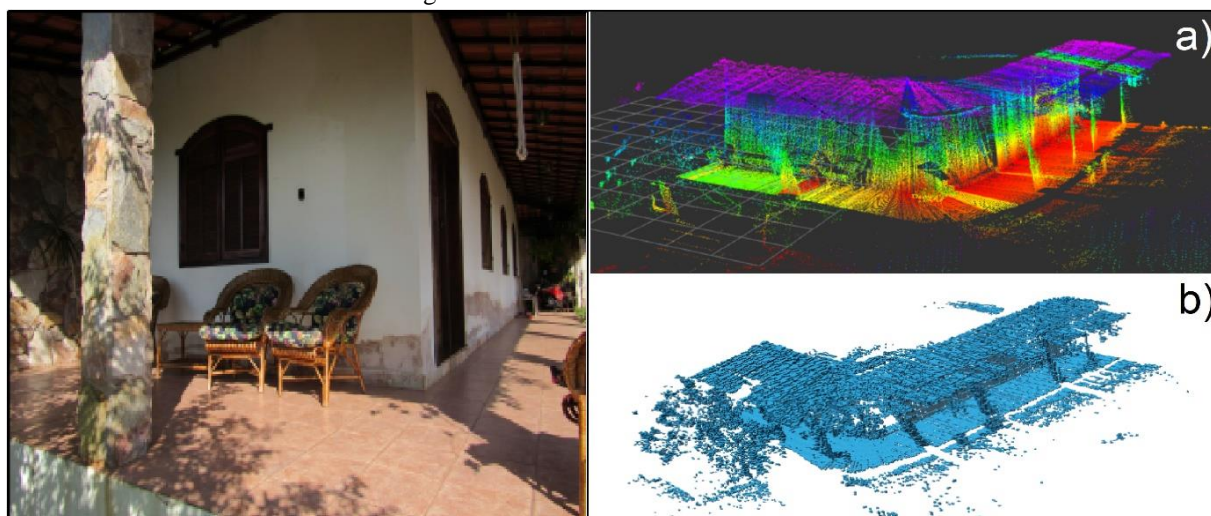
Quadro 2 – Mapeamento 3D de um ambiente interno

Referência	Rossi, Freitas e Reis (2019)
Objetivos	O trabalho tem como objetivo desenvolver um protótipo robótico com o sensoriamento em 3D feito com base em um sensor LiDAR, capaz de efetuar o mapeamento em ambientes simulados e reais
Principais funcionalidades	Funciona com o princípio de triangulação, que é uma metodologia empregada para medir distâncias de um objeto. Para funcionar dessa maneira, o sensor emite um feixe de laser que é refletido e captado por uma lente fotossensível. Dessa forma, quando o objeto a ser mapeado está mais próximo ou mais distante, os feixes tocam em uma parte diferente da lente e pelo princípio da triangulação geométrica é estimada a posição do objeto.
Ferramentas de desenvolvimento	Foi utilizado o sensor LiDAR modelo YDLIDARX4 para emitir o laser. Uma placa de desenvolvimento ESP32 com antena integrada. Uma Unidade Inercial de Medida (IMU) que é composto por um Acelerômetro e um Giroscópio MPU-6050. E um Encoder Rotativo KY-040.
Resultados e conclusões	Os resultados obtidos mostraram que a precisão de ambos os modelos é maior que 0,70, a área sob a curva (AUC) é maior que 0,80, e o modelo gerado a partir dos dados LiDAR é mais preciso.

Fonte: elaborado pelo autor.

Na Figura 4 são ilustrados os resultados capturados de uma varanda, que segundo Rossi, Freitas e Reis (2019), (a) é uma nuvem de pontos gerada através dos dados do sensor LiDAR com base nos movimentos de rotação e translação, (b) é um mapa desse mesmo ambiente gerado pela grade de ocupação tridimensional de uma biblioteca, chamada de OctoMap, com base na nuvem de pontos.

Figura 4 – Resultados obtidos do sensor



Fonte: Rossi, Freitas e Reis (2019).

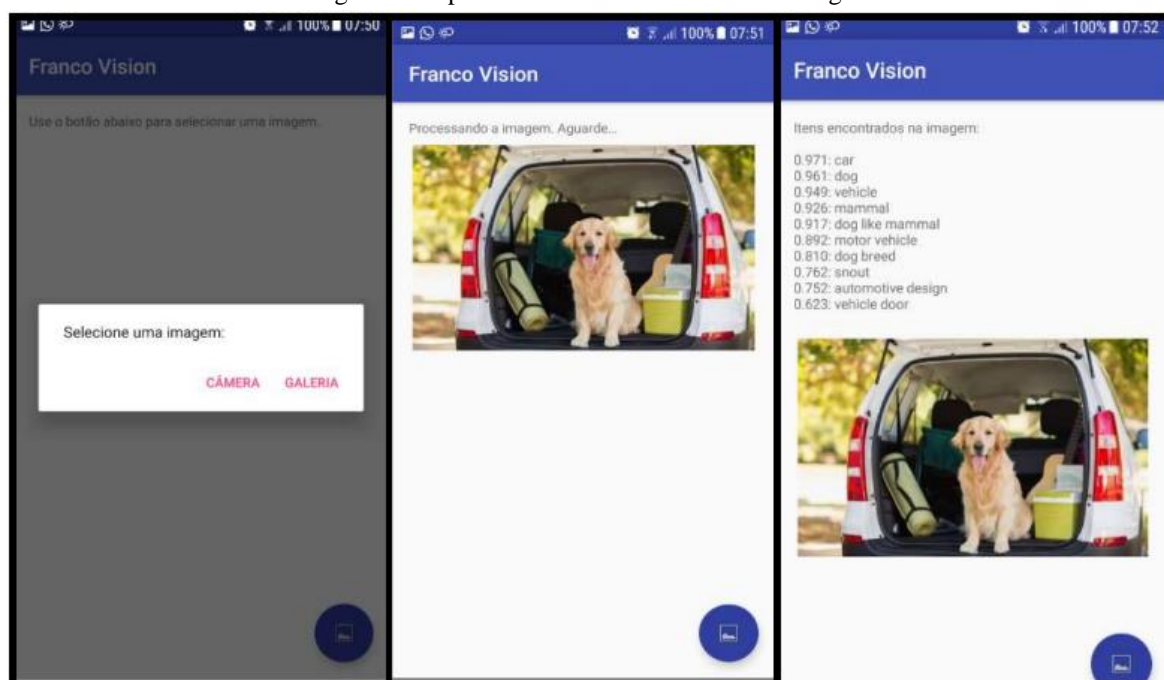
Quadro 3 – Aplicativo de reconhecimento de objetos

Referência	Franco (2018)
Objetivos	O objetivo era criar um aplicativo para instruir usuários, principalmente com deficiência visual, por meio da sintetização de voz em cada etapa do seu aplicativo de reconhecimento de objetos.
Principais funcionalidades	Instruções de voz para orientação e navegação pelo sistema. Reconhecer um objeto de uma foto da galeria ou do <i>feed</i> da câmera.
Ferramentas de desenvolvimento	Cloud Text-to-Speech para a sintetização de voz. API Cloud Vision para o reconhecimento de objetos.
Resultados e conclusões	O aplicativo não estava 100% apto para pessoas com deficiência visual. Dependendo de alguns fatores como luz, distância e outros ruídos o resultado do reconhecimento é excelente, mas quando a distância é incrementada a sua precisão começa a declinar.

Fonte: elaborado pelo autor.

Na Figura 5 é apresentado um breve comportamento do aplicativo de reconhecimento de imagens mencionado anteriormente, onde o usuário anexou uma imagem da galeria e a API retornou possíveis objetos que se encontram na imagem.

Figura 5 – Aplicativo de reconhecimento de imagem



Fonte: Franco (2018).

Quadro 4 – Comparação de dados derivados de LiDAR e VANT

Referência	Pereira (2022)
Objetivos	O objetivo é comparar mapas de suscetibilidade a deslizamentos gerados pelo algoritmo de aprendizado de máquina floresta aleatória (Random Forest - RF) com dados de LiDAR e VANT
Principais funcionalidades	Utilizar o LiDAR e VANT para extrair métricas como a Precisão, Acurácia, Recall e F1-Score (medida de precisão) e comparar os dois modelos de mapeamento de suscetibilidade a deslizamentos utilizando o RF.
Ferramentas de desenvolvimento	Modelo Digital de Elevação, Modelos Digital de Terrenos, sensor LiDAR, Veículo Aéreo Não-Tripulado (VANT)
Resultados e conclusões	Os resultados obtidos mostraram que a precisão de ambos os modelos é maior que 0,70, a área sob a curva (AUC) é maior que 0,80, e o modelo gerado a partir dos dados LiDAR é mais preciso.

Fonte: elaborado pelo autor.

Os trabalhos correlatos tiveram um papel importante principalmente na parte de entendimento do LiDAR e os possíveis problemas que podem aparecer, seja de precisão, jeito de utilização ou outro impedimento. Como por exemplo, o trabalho de comparação de dados derivados de LiDAR e VANT da Pereira (2022) mostrou que a precisão dos dados do LiDAR é superior ao do VANT em diversos cenários. O projeto de Rossi, Freitas e Reis (2019), permitiu entender melhor o funcionamento do sensor LiDAR e como utilizá-lo em cenários menores, além de dar sugestões de tratamento de ruído.

3 DESCRIÇÃO DO APLICATIVO

Para o desenvolvimento e teste da aplicação se utilizou o smartphone iPhone 13 Pro com o iOS 16.0, mas qualquer versão ou modelo de dispositivos móvel que seja superior e contenha o sensor LiDAR deve suportar a aplicação desenvolvida. Para realizar o desenvolvimento, foi utilizado um MacBook Pro com 8Gb de RAM, com processador i3 2.33Ghz, sem placa gráfica e um SSD.

A seguir na subseção 3.1 é apresentada a especificação, já na subseção 3.2 é abordado detalhes da implementação e funcionalidade do projeto.

3.1 ESPECIFICAÇÃO


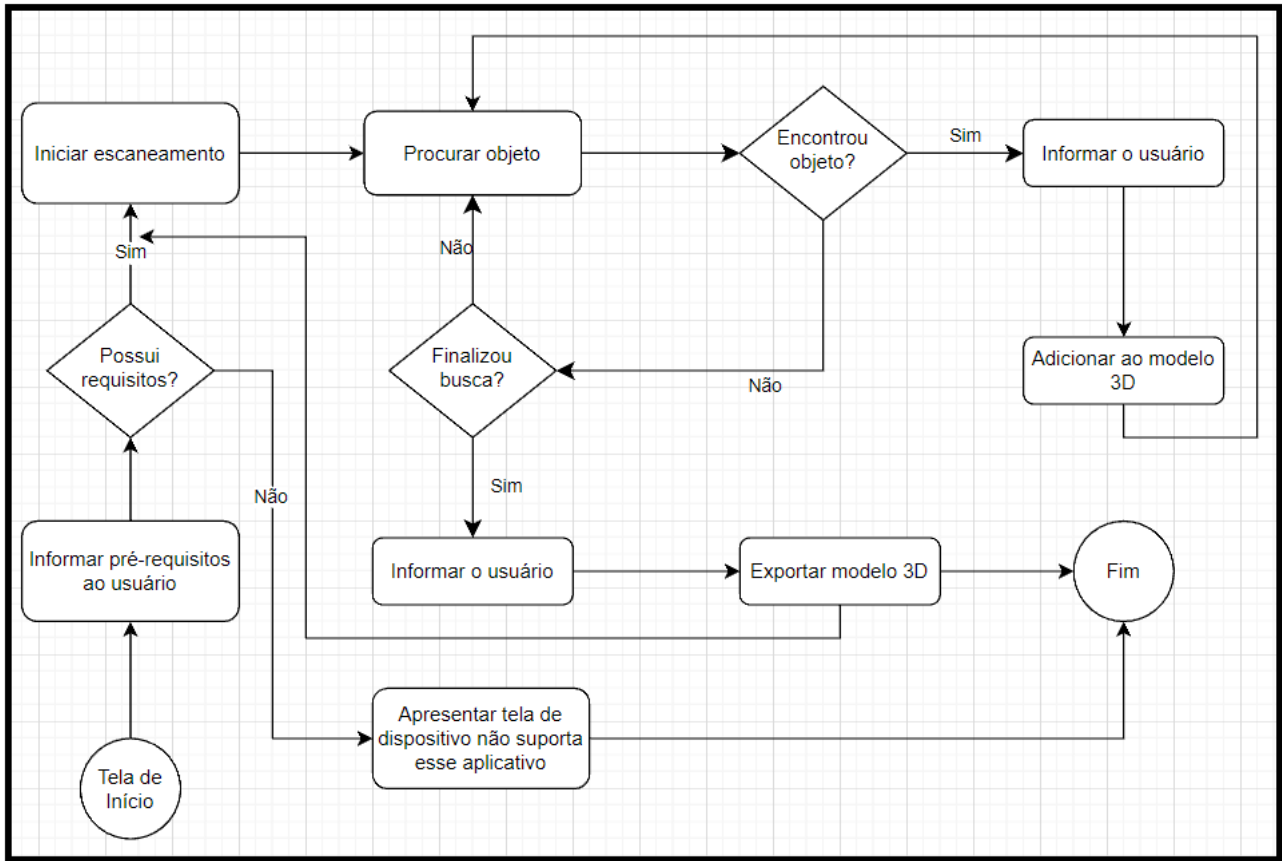
O fluxograma e o diagrama  foram construídos utilizando a ferramenta de trabalho Draw.IO, disponível gratuitamente na internet. O fluxograma de comportamento é representado na Figura 6 e se inicia informando ao usuário deficiente visual o que ele deve falar para conseguir interagir com o aplicativo e como iniciar o escaneamento. Ao sinalizar o início da busca do ambiente, o aplicativo abrirá a câmera e orienta o usuário que ele movimente o celular para cima e para baixo para uma melhora de captura do cenário atual, sempre buscando uma parede como ponto de partida. Com o escaneamento já em andamento, o aplicativo procurará novos objetos no cenário, ao encontrar um novo objeto ele informará ao usuário por comando de voz, adicionará o objeto encontrado ao modelo 3D do ambiente interno e seguirá com o escaneamento do ambiente. Se o aplicativo não encontrar nenhum objeto novo após alguns segundos, o escaneamento é finalizado e apresentado a tela que permite a exportação do modelo 3D gerado. O usuário pode finalizar o escaneamento a qualquer momento que ele desejar. Na tela de exportação do modelo 3D é possível voltar ao início e recomençar o escaneamento do ambiente. Se ao iniciar o aplicativo o usuário não possuir o sensor LiDAR, será apresentado uma tela de aviso informando ao usuário que ele não consegue executar o aplicativo.

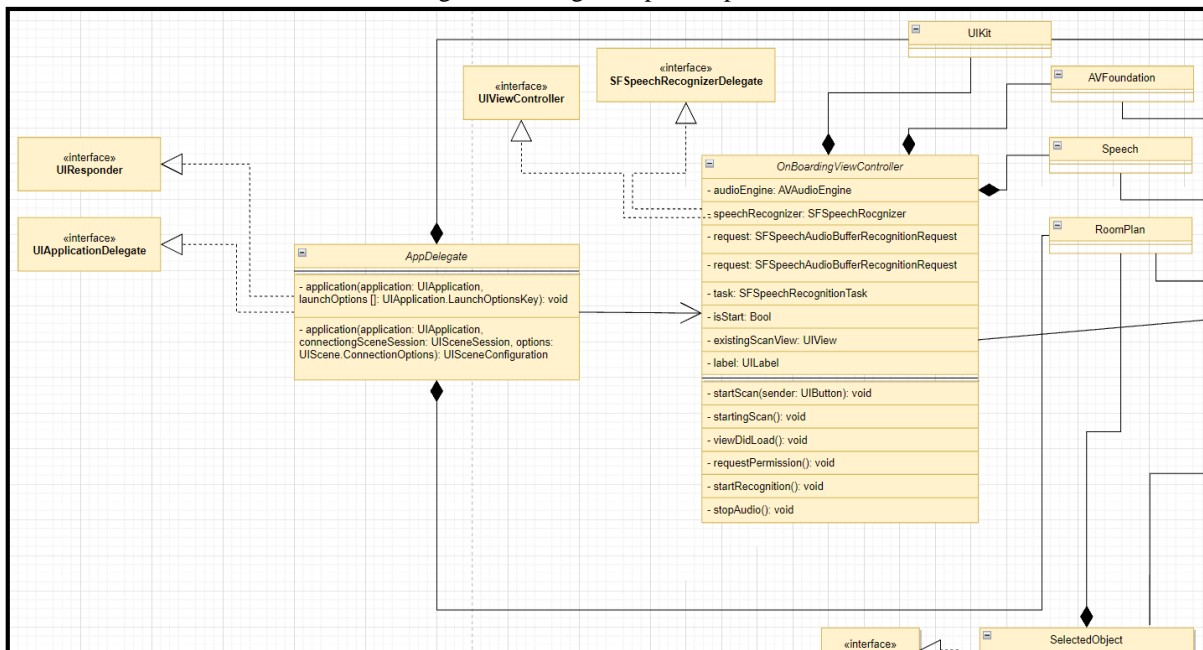
Figura 6 – Fluxograma de telas do aplicativo



Fonte: Elaborado pelo autor.

Na Figura 7 e Figura 8 é apresentado o Diagrama de Classes do aplicativo, onde mostra todas as classes, variáveis, métodos, agregações, composições e implementações de interface. As figuras foram divididas para se obter uma melhor compreensão e visualização do diagrama.

Figura 7 – Diagrama parcial parte 1



Fonte: Elaborado pelo autor.

```

classDiagram
    class RoomCaptureViewController {
        -exportButton: UIButton
        -doneButton: UIBarButtonItem
        -cancelButton: UIBarButtonItem
        -isScanning: Bool
        -roomCaptureView: RoomCaptureView
        -roomCaptureSessionConfig: RoomCaptureSession.Configuration
        -finalResults: CapturedRoom
        +listObjects []: SelectedObject
        -audioEngine: AVAudioEngine
        -speechRecognizer: SFSpeechRecognizer
        -request: SFSpeechRecognizer
        -task: SFSpeechRecognitionTask
        -synthesizer: AVSpeechSynthesizer
        -cancelScan []: String
        -doneScan []: String
        -startSession(): void
        -stopSession(): void
        -viewDidLoad(): void
        -viewWillAppear(animated: Bool)
        -viewWillDisappear(flag: Bool): void
        -setupRoomCaptureView(): void
        -captureSession(session: RoomCaptureSession, room: CapturedRoom): void
        -captureView(roomDataForProcessing: CapturedRoomData, error: Error): Bool
        -captureView(processedResult: CapturedRoom, error: Error): void
        -verifyObjectInMLKit(listObjects []: CapturedRoom.Object): void
        -openCameraAndGetSnapshot(): UIImage
        -identifyObjectAndReturnLabel(): String
        -verifyIfAlreadyHasObject(listObjects []: CapturedRoom.Surface, legend: String): void
        -sendTextToSpeech(text: String): void
        -listenSpeechToText(text []: String)
        -stopListen(): void
        -doneScanning(sender: UIBarButtonItem): void
        -cancelScanning(sender: UIBarButtonItem): void
        -exportResults(sender: UIButton): void
        -setActiveNavBar(): void
        -setCompleteNavBar(): void
    }

    class SelectedObject {
        +id: UUID
        +type: CapturedRoom.Surface.Category
        +legend: String
    }

    class RoomPlan {
    }

    class Speech {
    }

    class AVFoundation {
    }

    class UIKit {
    }

    class Foundation {
    }

    class RoomCaptureViewDelegate {
    }

    class RoomCaptureSessionDelegate {
    }

    RoomCaptureViewController --> SelectedObject
    RoomCaptureViewController --> RoomPlan
    RoomCaptureViewController --> Speech
    RoomCaptureViewController --> AVFoundation
    RoomCaptureViewController --> UIKit
    RoomCaptureViewController --> RoomCaptureViewDelegate
    RoomCaptureViewController --> RoomCaptureSessionDelegate
    Foundation --> SelectedObject
    
```

The diagram illustrates the architecture of the RoomCaptureViewController. It is a UIViewController that manages a RoomCaptureView and interacts with various system frameworks and interfaces. Key components include:

- RoomCaptureViewController**: The main controller class, which is a UIViewController. It manages a RoomCaptureView, a RoomCaptureSession, and a RoomCaptureSessionConfig. It also handles audio processing (AVAudioEngine, SFSpeechRecognizer, AVSpeechSynthesizer) and speech recognition tasks (SFSpeechRecognitionTask). It includes methods for starting/stopping sessions, capturing data, and handling UI events (button clicks, scanning, etc.).
- SelectedObject**: A class representing a selected object, with attributes: id (UUID), type (CapturedRoom.Surface.Category), and legend (String).
- RoomPlan**: A class representing a room plan.
- Speech**: A class representing speech-related data or actions.
- AVFoundation**: A framework used for audio processing.
- UIKit**: A framework used for UI elements and views.
- RoomCaptureViewDelegate**: An interface for the RoomCaptureView to communicate with the controller.
- RoomCaptureSessionDelegate**: An interface for the RoomCaptureSession to communicate with the controller.
- Foundation**: A framework used for object-oriented programming and data management.

Fonte: Elaborado pelo autor.

IMPLEMENTAÇÃO

Esta seção apresenta a arquitetura do aplicativo. O Quadro 5 apresenta os Requisitos Funcionais (RF), já o Quadro 6 apresenta os Requisitos Não Funcionais (RNF).

Esta seção apresen



Quadro 5 – Requisitos funcionais

Requisitos funcionais
RF01: o aplicativo deverá informar ao usuário quando há um novo obstáculo no visor da câmera
RF02: o aplicativo deverá permitir interação vocal
RF03: o aplicativo deverá criar uma planta em 3D do ambiente escaneado
RF04: o aplicativo deverá permitir exportar o modelo 3D do ambiente

Fonte: elaborado pelo autor.

Quadro 6 – Requisitos não funcionais

Requisitos não funcionais
RNF01: a detecção principal de objetos deve ser feita com a <i>framework</i> RoomPlan, quando não for possível, deve utilizar o ML Kit
RNF02: ao detectar um obstáculo, o aplicativo deve informar qual é o objeto encontrado

Fonte: elaborado pelo autor.

Para a implementação do aplicativo de orientação de pessoas com deficiência visual foi utilizado a linguagem de programação Swift, utilizando o XCode 14.0.1 e a versão compatível do MacOS 12.6. As bibliotecas utilizadas, como o AVFoundation, Speech, Room Plan, entre outros, estão em constantes atualizações e elas não possuem um controle de versionamento, além disso, é necessário garantir a última versão disponível para executar o aplicativo corretamente. O desenvolvimento foi dividido em 3 partes, 1) Inicialização do escaneamento do LiDAR; 2) Identificação dos objetos; 3) conversação com o aplicativo em todos os momentos.

Ao iniciar o aplicativo é necessário aprovar diversas permissões para que seja permitido o uso da câmera, microfone e alto falante. A inicialização do escaneamento é feita por meio da biblioteca Room Plan (ver subseção 2.1) com um evento privado nomeado `StartSession`, e após o disparo desse evento o LiDAR começa a funcionar. À medida que o LiDAR identifica novos objetos ele dispara um outro evento informando os objetos já identificados, e esse evento é chamado de `captureSession` com uma sobreposição específica que é o `didAdd`. Nesse `captureSession` é possível ver quais são todos os objetos já capturados pelo LiDAR, além disso o Room Plan faz uma triagem do objeto encontrado. A triagem é dividida em 5 grupos, sendo eles, portas, paredes, aberturas, janelas e objetos. No Quadro 7 é ilustrado o código do aplicativo onde é mencionado o evento do `captureSession` e os grupos de objetos recebidos.

Quadro 7 – Utilizando a biblioteca Room Plan

```

51
52     func captureSession(_ session: RoomCaptureSession, didAdd room: CapturedRoom) {
53
54         verifyIfAlreadyHasObject(listObjects: room.doors, legend: "porta")
55         verifyIfAlreadyHasObject(listObjects: room.walls, legend: "parede")
56         verifyIfAlreadyHasObject(listObjects: room.openings, legend: "passagem")
57         verifyIfAlreadyHasObject(listObjects: room.windows, legend: "janela")
58
59         verifyObjectInMLKit(listObjects: room.objects)
60
61     }
62

```

Fonte: Elaborado pelo autor.

No Quadro 7, o objeto retornado `room` possui vários vetores de Surface ou Object. Um Object ou Surface possui algumas propriedades como identificador (UUID), categoria, precisão, entre outras que não foram utilizados. Não é possível determinar a distância que o objeto se encontra da câmera, porque infelizmente a *framework* não disponibiliza essa propriedade.

Como o foco principal do Room Plan não é informar quais são os objetos encontrados no ambiente ele acaba generalizando apenas para objeto, e nesses casos se optou em utilizar o ML Kit (ver subseção 2.2). No escopo atual o Room Plan não consegue retornar a imagem do *feed* da câmera como um objeto utilizável, no caso o `CVPixelBuffer`, então é necessário utilizar a própria câmera para tirar uma foto desse instante que é encontrado o objeto e encaminhar essa imagem para o ML Kit para realizar a sua identificação. Após receber uma imagem da câmera no formato `UIImage` ela é encaminhada para o ML Kit para realizar a identificação do objeto, com o resultado da identificação, é informado ao usuário final em comando de voz. Na Quadro 8 é apresentado o trecho do código onde é recebido a `UIImage`, enviada para identificação e recebendo o resultado da identificação.

Quadro 8 – Trecho de código ML Kit

```

239
240 func detectLabels(image: UIImage?, shouldUseCustomModel: Bool) {
241     guard let image = image else { return }
242
243     var options: CommonImageLabelerOptions! = ImageLabelerOptions()
244     options.confidenceThreshold = NSNumber(floatLiteral: Constants.labelConfidenceThreshold)
245
246     let onDeviceLabeler = ImageLabeler.imageLabeler(options: options)
247
248     let visionImage = VisionImage(image: image)
249     visionImage.orientation = image.imageOrientation
250
251     weak var weakSelf = self
252     onDeviceLabeler.process(visionImage) { labels, error in
253         guard let strongSelf = weakSelf else {
254             print("Self is nil!")
255             return
256         }
257         guard error == nil, let labels = labels, !labels.isEmpty else {
258             let errorString = error?.localizedDescription ?? Constants.detectionNoResultsMessage
259             strongSelf.resultsText = "On-Device label detection failed with error: \(errorString)"
260             strongSelf.showResults()
261             return
262         }
263
264         strongSelf.resultsText = labels.map { label -> String in
265             return "Label: \(label.text), Confidence: \(label.confidence), Index: \(label.index)"
266         }.joined(separator: "\n")
267         strongSelf.showResults()
268     }
269 }

```

Fonte: Elaborado pelo autor.

Todos os botões do aplicativo podem ser interagidos por comandos de voz utilizando os *frameworks* Speech e AVFoundation. Esse procedimento ficou demasiado simples com um pequeno impedimento, no AVFoundation é possível determinar qual a língua a ser falada, já o Speech não permite isso de maneira tão simplificada, onde o padrão é o inglês. Não somente isso, o Speech entende diversas línguas ao mesmo tempo, ou seja, ele entende outros idiomas e escreve corretamente no seu *Buffer* de resposta a palavra em português, espanhol e afins, mas a língua que possui a maior precisão de entendimento é o inglês. Em função da língua predominante ser o inglês, ao entender uma palavra em outro idioma o Speech efetua a tradução para o inglês, resultando em uma dificuldade para extrair o idioma falado pelo usuário. No Quadro 9 é ilustrado o comportamento do Speech onde o usuário fala o comando *Iniciar busca*, como resultado a palavra *iniciar* é traduzida para o inglês da forma que é compreendida resultando em *inside* e *initiate*, já a palavra *busca* é permanecida em português.

Quadro 9 – Retorno do Buffer do microfone

```

to list voice folder
ACCEPTED
any
inside
inside busca
inside busca en
inside busca
inside busca en
inside busca en the shower
inside busca en the shower busca
inside busca in the shower busca
inside busca en isha busca initiate
inside busca en isha busca initiate busca

```

Fonte: Elaborado pelo autor.

4 RESULTADOS

Esta seção mostra os testes, **incitamentos** e resultados. Na subseção 4.1 é detalhado os desafios encontrados no desenvolvimento do projeto, juntamente com os testes realizados e na subseção 0 é apresentado os resultados obtidos.

4.1 TESTES E DESAFIOS

Para facilitar a utilização do sensor LiDAR, o desenvolvimento se iniciou com a biblioteca Room Plan desenvolvida em Swift. Ao mesmo tempo que ela facilita o acesso aos resultados em função da sua triagem, ela não permite determinar a distância entre o objeto e a câmera, causando um impedimento nessa métrica. Sem a distância exata do objeto, no momento de informar o usuário sobre o objeto, fica relativamente vago aonde o objeto se encontra, podendo atrapalhar a orientação do usuário.

O segundo desafio foi fazer a biblioteca de reconhecimento de voz, o AVFoundation, entender a língua nativa do usuário, isso porque ao entender uma palavra, internamente ela faz a tradução imediata para o inglês, se essa palavra existe no dicionário inglês então ela é traduzida, se não existe ela é permanecida na língua que foi compreendida. O Quadro 9 é o exemplo perfeito desse desafio, onde o usuário falava a frase *iniciar busca* e a *framework* traduzia apenas a primeira palavra para o inglês e a segunda era permanecida em português. Alguns tratamentos foram feitos no código para ele entender essa mistura de idiomas e mitigar esse problema de tradução, mas os problemas não são totalmente mitigados.

Já o terceiro foi encontrar uma Machine Learning que fazia integração com o iOS de maneira facilitada, isso porque os modelos de ML já existentes da Apple, o CoreModel e o CoreVision não atendiam ao escopo de objetos de uma casa. Mesmo o treinamento do modelo ser em Python, não era interessante fazer o treinamento desde o zero para atender todos os cenários de objetos que é possuído dentro de uma casa, sendo que já existem MLs treinadas para objetos do interior de uma casa. O ML Kit da Google atende muito bem aos requisitos de um ambiente interno, contudo as configurações necessárias para executá-lo **possuem conflito direto com** as configurações do Room Plan.

O aplicativo foi testado com uma pessoa que não possui deficiência visual, para tentar ser mais fidedigno ao público-alvo foi colocado uma venda sob os olhos da usuária. Ela utilizou o aplicativo em dois cenários, no primeiro cenário a usuária visualizou a sala e tinha noção de espaço, saída, disposição de cada objeto da sala e onde ela estava localizada no cenário, e onde estava a porta de saída. O local de testes foi em uma sala do segundo andar do Bloco S na Universidade Regional de Blumenau (FURB). Na Figura 9 é mostrado o cenário real onde a usuária foi inserida e a sua localização inicial no cenário.

Figura 9 – Localização da usuária no primeiro cenário



Fonte: Elaborado pelo autor.

Ao iniciar o aplicativo e começar a se locomover, já havia uma mesa com algumas cadeiras em sua frente e o seu objetivo era chegar até a porta e andar pelo corredor da FURB sem bater nos obstáculos no meio do caminho.

No segundo cenário, a sala foi modificada e a usuária ficou vendada o tempo todo durante as modificações da sala. Além disso, ela foi inserida em um ponto de partida diferente do anterior, fazendo com que ela perdesse a noção de espaço da sala. O objetivo dela continuou o mesmo, sair da sala pela porta evitando os obstáculos do cenário. Na Figura 10 é possível visualizar o segundo cenário de teste com as alterações do cenário.

Figura 10 – Localização da usuária no segundo cenário



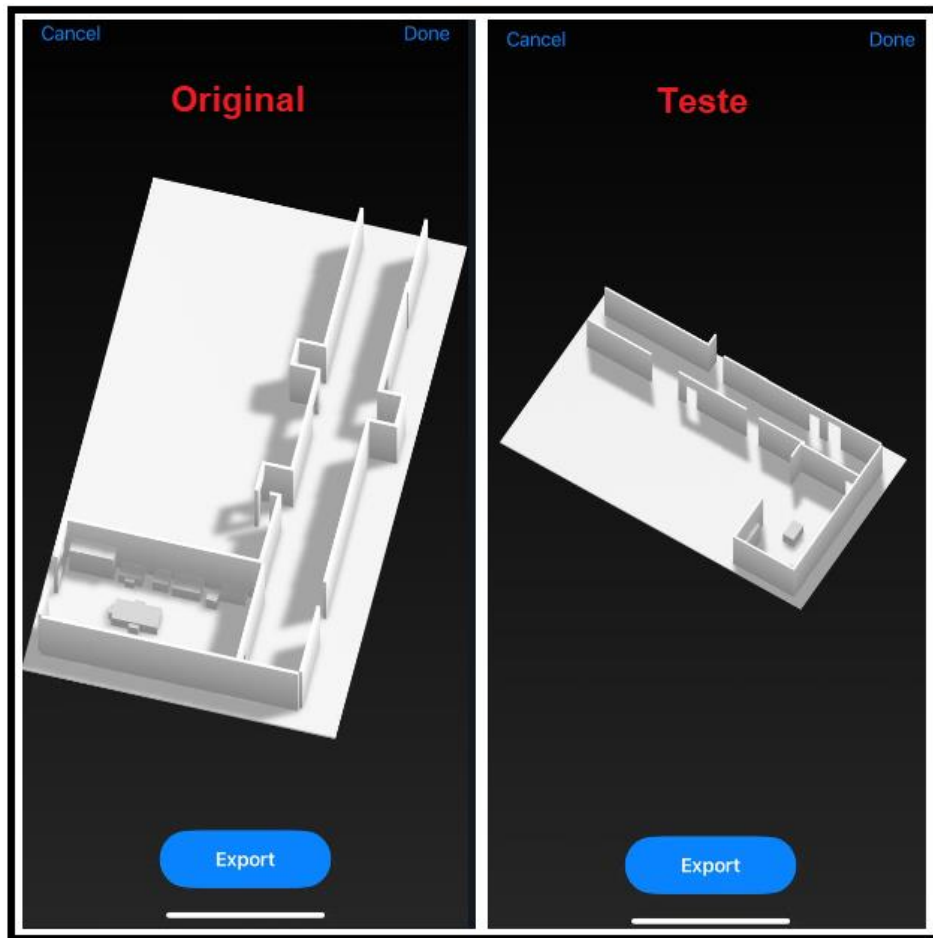
Fonte: Elaborado pelo autor.

4.2 RESULTADOS OBTIDOS

No primeiro cenário de testes a usuária conseguiu locomover-se com facilidade, levando em conta que ela possuía noção e conhecimento do espaço. O sensor LiDAR conseguiu capturar quase todo o cenário que ela estava inserida, isso porque para o LiDAR conseguir escanear corretamente, ele necessita de alguns poucos segundos para fazer a identificação do objeto e passar o retorno para o usuário, mas durante a utilização havia momentos em que o celular era movimentado diversas vezes em um curto período, fazendo com que ele não capturasse corretamente todas as informações do cenário. Na Figura 11 é mostrado a comparação de resultados do primeiro cenário, onde no lado esquerdo da imagem está o que deveria ter sido mapeado e do lado direito o que foi mapeado durante o teste da aplicação.

Além de não capturar adequadamente os objetos do interior da sala, como mesas, cadeiras e janelas, durante o percurso do corredor não foi possível identificar corretamente as portas e depois de alguns instantes caminhando, o aplicativo perdeu a localização da última parede, fazendo assim com que ele não conseguisse continuar o mapeamento, ou seja, o aplicativo parou de identificar novos obstáculos e por consequência não informando mais nada para o usuário. Nessa situação, existem duas formas de resolver esse problema, a primeira é fazer com que o usuário volte com o aplicativo até o ponto onde ele se perdeu e continuar o escaneamento a partir deste ponto, que para uma pessoa com deficiência visual isso não é uma maneira prática de solucionar o problema, já a segunda forma é cancelando o escaneamento e iniciando uma nova busca do zero.

Figura 11 – Comparação de resultado do primeiro cenário



Fonte: Elaborado pelo autor.

O segundo cenário de testes, que envolve alterações no cenário e removendo o senso de localização da usuária, teve resultados melhores do que o primeiro cenário na questão da orientação do usuário, porém nesse caso o sensor LiDAR não conseguiu identificar um objeto importante nesse escopo. Na Figura 10 é possível visualizar uma escada que possui dois degraus próximo a saída da sala. Não foi possível mapear esse objeto nem no cenário original, que por algum motivo o sensor não consegue fazer a identificação dele como um objeto. Na Figura 12 é possível visualizar esse comportamento nitidamente durante o uso do aplicativo, a câmera estava focada na escada em diversos ângulos e ainda assim não foi realizado a identificação do objeto.

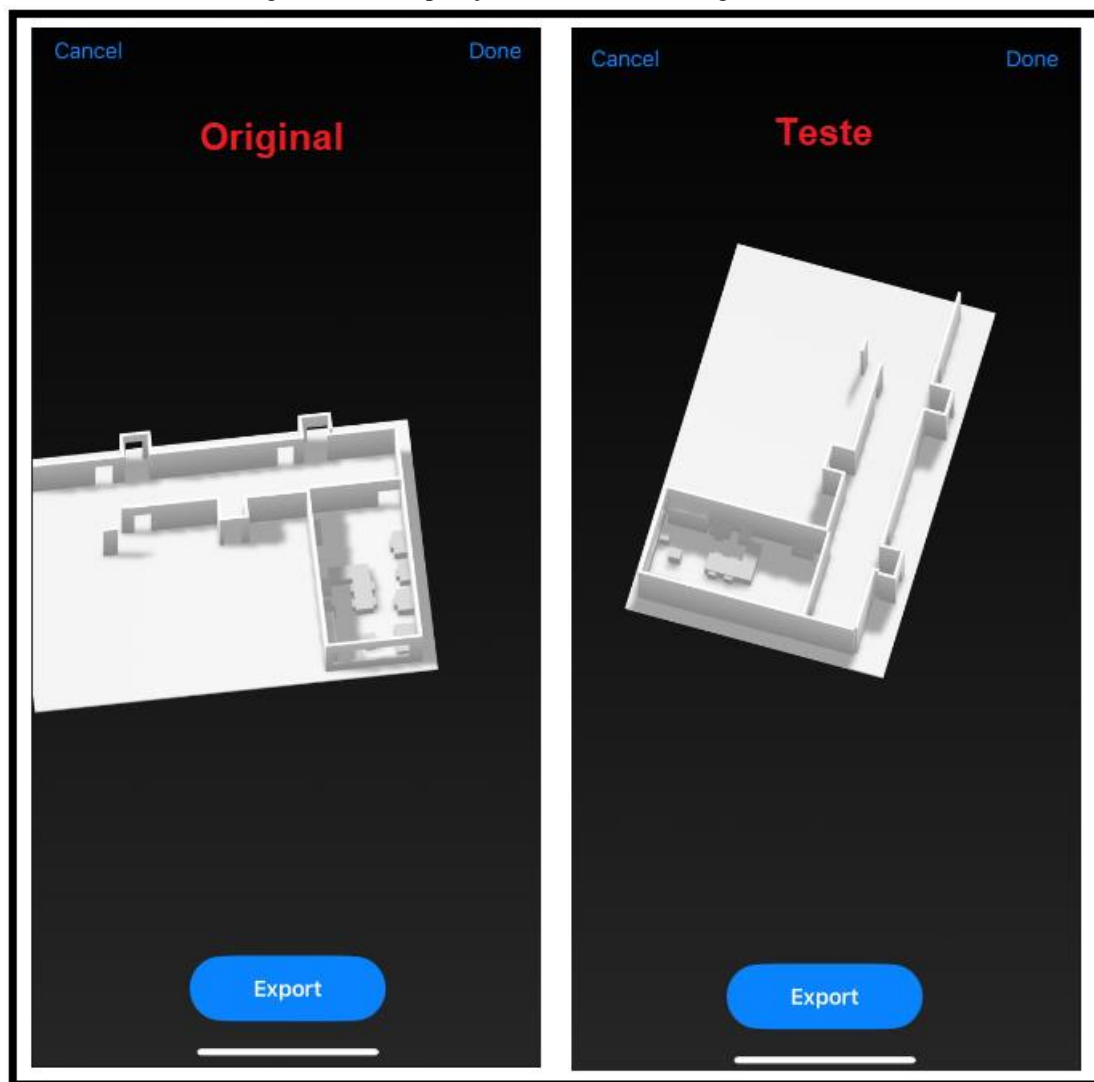
Figura 12 – Falha durante a identificação da escada



Fonte: Elaborado pelo autor.

Como constatado anteriormente, com a exceção da escada, o escaneamento do segundo cenário de testes teve resultados bem melhores, porém a usuária constatou que durante a sua movimentação mais a soma de vários objetos no cenário ao mesmo tempo, faz com que o aplicativo informe diversas vezes que há um obstáculo em seu caminho, prejudicando o entendimento de noção de localização do objeto e a velocidade de fala do aplicativo é muito acelerada. “Fazer a fala do aplicativo um pouco mais devagar ao identificar os obstáculos e objetos. Também a voz do aplicativo é um pouco baixa.”. Na Figura 13 é realizado um comparativo do segundo cenário entre o que deveria ser mapeado e o que foi mapeado pela usuária durante a utilização do aplicativo. É perceptível que os resultados foram bem semelhantes, inclusive na situação que a escada não é detectada. A usuária comentou que o aplicativo trás segurança para o usuário, de não ter perigo de se machucar em algum móvel ou obstáculo na sua frente.

Figura 13 – Comparação de resultados do segundo cenário



Fonte: Elaborado pelo autor.

5 CONCLUSÕES

☞ Ao efetuar os testes foi visto que o aplicativo atingiu o objetivo principal que é informar o usuário que há um obstáculo no seu campo de visão e que ele está próximo, porém o primeiro objetivo específico de determinar a distância do objeto não foi alcançado devido as limitações da biblioteca utilizada. O segundo objetivo específico foi atendido porque a comunicação do aplicativo com o usuário atendia as necessidades, permitindo que ele saiba exatamente em qual tela que ele se encontra e informando em todos os momentos que um objeto novo é encontrado. Em contrapartida foi insuficiente nas limitações da biblioteca ao tentar traduzir todas as palavras do português para o inglês.

Apesar dos resultados obtidos, durante a utilização do ML Kit há vários fatores que influenciam seu resultado, sendo os principais a iluminação ☞ o objeto encontrado não é o objeto principal do cenário, causando assim uma pequena diferenciação de qual objeto ele informa ao usuário. Considerando que a comunicação por voz é uma tecnologia que existe há um determinado tempo, ela ainda não está preparada para todos os idiomas, atualmente a que está funcionando bem é apenas para o inglês. Com isso, esse aplicativo não pode ser utilizado em situações reais por deixar o usuário sem noção exata de onde o objeto se encontra, entretanto, este aplicativo pode engajar trabalhos futuros nessa linha de pesquisa. As possíveis propostas de extensão são: utilizar o modelo 3D para alguma funcionalidade como a localização indoor utilizando Beacons; melhorar a utilização do LiDAR com o objetivo de trazer a localização exata do objeto; utilizar outras bibliotecas para o reconhecimento de voz permitindo que o usuário se comunique apenas com o português.

REFERÊNCIAS

- APPLE. **AVFoundation**. [2022a] Disponível em: <https://developer.apple.com/documentation/avfoundation>. Acesso em: 23 nov. 2022.
- APPLE. **RealityKit**. [2022b]. Disponível em: <https://developer.apple.com/documentation/realitykit>. Acesso em: 23 nov. 2022.
- APPLE. **RoomPlan**. [2022c]. Disponível em: <https://developer.apple.com/documentation/roomplan>. Acesso em: 04 nov. 2022.
- APPLE. **Speech**. [2022d]. Disponível em: <https://developer.apple.com/documentation/speech/>. Acesso em: 10 nov. 2022.
- CAMBRAIA, Maria Izabel de Albuquerque; NAZIMA, Maira Tiyomi Sacata Tongu. Avaliação dos sintomas de depressão em pessoas com deficiência visual. **Rev. SBPH**, São Paulo, v. 24, n. 1, p. 79-90, jun. 2021. Disponível em: http://pepsic.bvsalud.org/scielo.php?script=sci_arttext&pid=S1516-08582021000100008&lng=pt&nrm=iso. Acesso em: 24 nov. 2022.
- CASTRO, Oliveiros Barone. **Relações entre percepção auditiva e orientação e mobilidade em um grupo de pessoas com deficiência visual usuárias de cão**. 2019. 117 f. Dissertação (Mestrado) - Curso de Fonoaudiologia, Pontifícia Universidade Católica de São Paulo, São Paulo, 2019. Disponível em: <https://tede.pucsp.br/bitstream/handle/22132/2/Oliveiros%20Barone%20Castro.pdf>. Acesso em: 24 nov. 2022.
- FRANCO, Jean Carlos. **Aplicativo para o reconhecimento de objetos em imagens utilizando a API Cloud Vision destinado a pessoas portadoras de deficiência visual**. 2018. 56 f. TCC (Graduação) - Curso de Sistemas de Informação, Ciências Exatas e Naturais, Universidade Regional de Blumenau (Furb), Blumenau, 2018. Disponível em: <http://dsc.inf.furb.br/tcc/index.php?cd=11&tcc=1961>. Acesso em: 27 mar. 2022.
- FREITAS, Daniela Gonçalves da Silveira. **Orientações para interação com deficientes visuais e auditivos**. IFBA - Instituto Federal de Educação, Ciência e Tecnologia da Bahia, [S. L.], 2018. Disponível em: <https://portal.ifba.edu.br/conquista/napnee/paginas-links/orientacoes-para-interacao-com-deficientes-visuais-e-auditivos#wrapper>. Acesso em: 23 nov. 2021.
- GOOGLE. **Kit de ML**. 2022. Disponível em: <https://developers.google.com/ml-kit/guides>. Acesso em: 27 nov. 2022.
- GOULART, Gielez Feldhaus. **Aplicativo para auxiliar crianças autistas no desenvolvimento e aquisição da linguagem**. 2016. 50 f. TCC (Graduação) - Curso de Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2016.
- HERCHONVICZ, Andrey L.; FRANCO, Cristiano R.; JASINSKI, Marcio G.. A comparison of cloud-based speech recognition engines. In: **COMPUTER ON THE BEACH**, 10., 2019, Florianópolis. **Anais [...]**. Florianópolis: Computer On The Beach, 2019. p. 366-375. Disponível em: <https://periodicos.univali.br/index.php/acotb/article/view/14332>. Acesso em: 24 nov. 2022.
- PEREIRA, Felicia França. **Comparação de dados derivados de LiDAR e VANT para o mapeamento de suscetibilidade de deslizamento usando o algoritmo Random Forest**. 2022. 21 f. Dissertação (Mestrado) - Curso de Desastres Naturais, O Instituto de Ciência e Tecnologia, Universidade Estadual Paulista (Unesp), São José dos Campos, 2022. Disponível em: <https://repositorio.unesp.br/handle/11449/237072>. Acesso em: 18 nov. 2022.
- ROSSI, Túlio Xavier; FREITAS, Elias José de Rezende; REIS, Agnaldo José da Rocha. **Mapeamento Tridimensional de Ambientes Internos Utilizando um Sensor LIDAR**. 2019. 62 f. Monografia (Especialização) - Curso de Engenharia de Controle e Automação, Cecaui, Universidade Federal de Ouro Preto -Ufop, Ouro Preto, 2019. Disponível em: <https://monografias.ufop.br/handle/35400000/2439>. Acesso em: 24 set. 2021.
- SILVA, William Lopes da. **Black Glasses: assistente para deficientes visuais via geolocalização**. 2019. 20 f. TCC (Graduação) - Curso de Bacharel em Ciência da Computação, Universidade Regional de Blumenau (FURB), Blumenau, 2019. Disponível em: http://dsc.inf.furb.br/arquivos/tccs/monografias/2019_2_william-lopes-da-silva_monografia.pdf. Acesso em: 22 jun. 2022.