DESENVOLVIMENTO DE UMA BIBLIOTECA PARA O USO DO SENSOR LIDAR EM DISPOSITIVOS IOS

Gabriel Luís Fernando Vieira de Souza, 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

glfsouza@furb.br, dalton@furb.br

Resumo: *O LiDAR, um sensor de medição de distância, é uma funcionalidade inovadora trazida para dispositivos com sistema iOS que contribui no mapeamento de grandes ambientes em frações de minutos, permitindo a digitalização desses ambientes em modelos 3D. Sua importância é evidenciada em sua aplicação na área florestal, devido sua performance e custo-benefício, assim como a importância da digitalização tridimensional é destacada no seu intuito de preservação histórica. Assim sendo, o presente estudo tem como objetivo geral criar uma biblioteca que permita que uma aplicação digitalize objetos reais em objetos que possam ser manipulados em um ambiente 3D virtual utilizando LiDAR dos dispositivos iPad Pro da Apple, por meio de uma revisão bibliográfica e testagem da capacidade do dispositivo de fazer a digitalização em diversos cenários, resultando no entendimento da utilização referente a medição de distância e digitalização 3D e identificação de limitações da funcionalidade, contribuindo para estudos futuros no campo.*

Palavras-chave*: LiDAR. iOS. Digitalização. 3D. Biblioteca.*

1 INTRODUÇÃO

Em um mundo globalizado e digitalmente conectado nos deparamos com possibilidades que nos permitem registrar um momento para que ele seja recordado posteriormente de maneira cada vez mais tecnológica. Diariamente se fotografa ou se filma cenas, seja para compartilhar uma viagem em uma rede social, guardar uma recordação especial ou até mesmo registrar um quadro na sala de aula para não esquecer o seu conteúdo. Com essa premissa, as empresas de tecnologia para dispositivos móveis estão constantemente inovando na forma como utilizamos ferramentas de filmagem e fotografia, introduzindo novas funcionalidades, que vão além de poder utilizar *zoom* nos registros dos momentos, até a possibilidade de efeito panorâmico na captura das imagens. Uma dessas inovações, anunciada primeiramente no iPad Pro da Apple, em 2020, e, posteriormente no iPhone 12 Pro, no mesmo ano, foi o LiDAR, sigla para Light Distance And Ranging. O LiDAR consiste em "um método de sensoriamento ativo que pode precisamente medir distâncias, transmitindo energia laser e analisando a energia retornada" (BAUWENS *et al*., 2016 p. 2, tradução nossa).

Historicamente esta tecnologia já vem sendo utilizada antes mesmo do interesse da Apple, principalmente na área florestal. Giongo *et al.* (2010) explicam que o LiDAR tem muitas vantagens para fazer uma digitalização em comparação com imagens de satélite para um mapeamento florestal, visto que não depende da luz solar, fazendo assim que seja mais fácil ignorar sombras ocasionadas pelas nuvens, assim como seus feixes de laser que são capazes de penetrar as copas das árvores facilitando na descoberta do relevo do terreno nas florestas. Outra aplicação foi utilizada com o sensor Zephir LiDAR, no ano de 2017, para construir um medidor de velocidade do vento para usinas eólicas pelos pesquisadores Nassif, Passos e Pimenta (2017), que enfatizaram que o "LiDAR é um sistema confiável, robusto e a custos relativamente baixos" (NASSIF; PASSOS; PIMENTA, 2017, p. 74), que favorece a adoção nas mais diversas áreas, destacando a performance e o custo-benefício da tecnologia. Para dispositivos móveis, a principal funcionalidade do sistema é perceber profundidade através da distância em um ambiente e mapear o relevo de objetos.

Outro uso da tecnologia LiDAR é na área de digitalização tridimensional. Gonzo *et al.* (2007) elencam que a digitalização 3D, é o processo de utilizar de várias fontes como escaneamento terrestre e fotografia, para criar um modelo em três dimensões. Os autores também descrevem a importância desse processo na preservação histórica, exemplificando a catalogação de monumentos em repositórios para a documentação e visualização remota desses monumentos. O LiDAR nesse âmbito, vem como apoio no mapeamento rápido de ambientes e a transformação desse mapeamento em um modelo 3D, já que ele é capaz de medir os arredores do ambiente e fazer a transformação em *mashes* (malhas 3D) que posteriormente formam o objeto tridimensional.

O LiDAR nos dispositivos Apple entra para agregar novas funcionalidades no ARKit, o framework de Realidade Aumentada (ou Agumented Reality – AR) do iOS. O ARKit se propõe a adicionar objetos virtuais 3D ou 2D através da visualização da câmera do dispositivo de uma forma que pareça que esses objetos estejam interagindo com o mundo real, conforme explica Apple (2022b). Uma dessas funcionalidades adicionadas no ARKit com o lançamento do LiDAR nos dispositivos da Apple foi a Depth API, que faz possível a captura da profundidade da cena pixel a pixel combinado com uma *mesh* 3D, gerando um mapeamento que pode ser utilizado para gerar medidas mais precisas de um ambiente ou na

geração de um Scene Geometry, que consiste em um mapa topológico do mundo real com categorizações de objetos também reais para serem interagíeis em uma aplicação que utiliza o ARKit.

Com o contexto descrito acima, o objetivo geral desse trabalho é criar uma biblioteca que permita que uma aplicação digitalize objetos reais em objetos que possam ser manipulados em um ambiente 3D virtual utilizando LiDAR dos dispositivos iPad Pro da Apple. Já os objetivos específicos são permitir o desenvolvimento de aplicações com digitalização 3D e construir abstrações à tecnologia LiDAR para a digitalização 3D com as bibliotecas do ARKit. Assim sendo, será realizada uma revisão bibliográfica consultando a documentação do ARKit e artigos sobre o assunto como base de estudo, bem como serão testadas as capacidades do dispositivo de fazer a digitalização de forma escalável contemplando diversos cenários para que sejam identificados os possíveis problemas em um uso cotidiano.

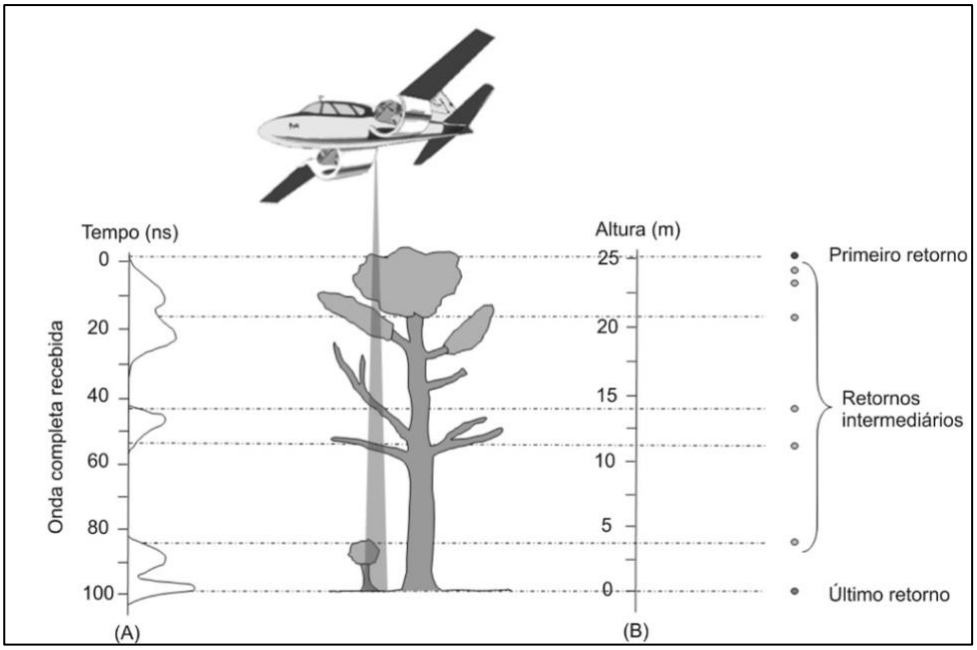
2 FUNDAMENTAÇÃO TEÓRICA

Para esse escrito, foi idealizado contextualizar a utilização do sensor LiDAR nas mais diversas áreas para digitalização 3D, categorizar os métodos de utilização desse sensor e consolidar como o LiDAR da Apple pode contribuir com essa área. Nesse intuito, a fundamentação teórica, foi dividida em três subtópicos. O primeiro se aprofundando mais da contribuição do LiDAR na área de digitalização 3D e o segundo trazendo como a Apple se preparou para a adição desse sensor em seus dispositivos iOS com o ARKit. Por fim, no terceiro, são trazidos trabalhos correlatos com o escrito atual.

2.1 DIGITALIZAÇÃO 3D

Nesta subseção será elencado formas de escaneamento utilizando LiDAR dentre eles o escaneamento aéreo, o escaneamento fixo e o escaneamento móvel. O escaneamento aéreo, como já foi trazido anteriormente, é utilizado, principalmente, na engenharia florestal para escaneamento de florestas. Como explica Muhadi (2020), se trata de um veículo aéreo que pode ser um helicóptero ou uma aeronave de asa fixa que carrega sensores e dentre eles o sensor LiDAR, que emite um feixe de laser em direção ao chão mapeando o que encontrar e o que for retornado ao sensor é gravado, em que, por exemplo, o tempo de viagem do laser é utilizado para medir a distância até o objeto. Na Figura 1 é possível ver um exemplo desse escaneamento que é utilizado na área florestal para o mapeamento do solo, onde pode-se observar que se trata de um processo em que o sensor, quando envia o feixe de luz para baixo, retorna informação da distância em metros calculada (Altura (m) na imagem) a cada n nano segundos (Tempo (ns) na imagem) que pode ser observado nas réguas a esquerda e a direita respectivamente. Cada linha pontilhada na horizontal representa os retornos de informação que o LiDAR traz, concluindo que para medir uma altura de 25 metros o sensor LiDAR utilizado pelos autores Giongo *et. al.* (2010) demoraria 100 nano segundos.

Figura 1 - Escaneamento Aéreo utilizando LiDAR.



Fonte: Giongo *et. al.* (2010).

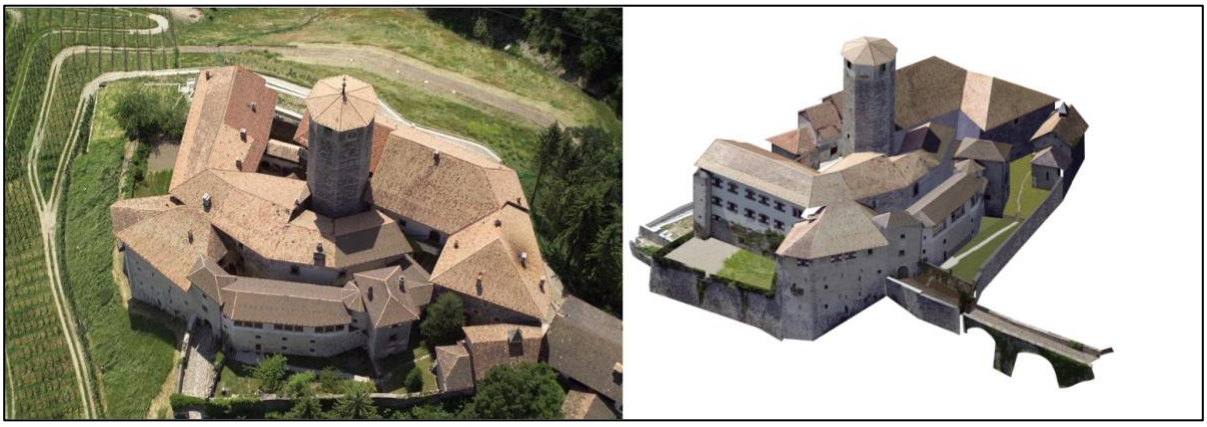
Nassif, Passos e Pimenta (2017) exemplificam o sensor fixo por meio de seu trabalho de utilização para medição de velocidade do vento, utilizando um sensor que "emite raio laser para o centro de um espelho que fica rotacionando e reflete o raio laser" (MUHADI *et. al.*, 2020) mapeando o ambiente. Por fim, o móvel, como enfatizam Muhadi *et. al.* (2020), é muito parecido com o escaneamento aéreo, pois digitaliza apenas uma direção e o portador do sensor se move

para mapear o ambiente que geralmente é combinado com um sensor de Global Position System (GPS), responsável por fazer as correções necessárias devido o movimento. No caso dos dispositivos Apple, a forma utilizada é o sensor móvel e como é mencionado por Apple (2022b), esse sensor em conjunto com o ARKit ajuda a digitalizar formas do mundo real transformando as coordenadas da distância em vértices que formam um *mash* a partir dos polígonos formados com a ligação desses vértices.

LiDAR, é uma tecnologia promissora que com sua adição em dispositivos móveis se torna mais atraente ao consumidor final. Na área florestal, como indagam Bauwens *et al*. (2016), já é estudada a possibilidade da utilização de sensores LiDAR portáteis, pois a metodologia de mapeamento aéreo não é capaz de destacar todos os elementos das florestas, onde um mapeamento de campo poderia resultar em um detalhamento mais abrangente das árvores. Com a chegada desse sensor nos dispositivos Apple, caso a tecnologia se prove eficiente, esta pesquisa poderia ser feita até mesmo por um aparelho celular.

Além da área florestal, a digitalização tridimensional é muito útil para preservação histórica. No trabalho de Gonzo *et al.* (2007) foram utilizados diversos métodos de modelagem 3D com o intuito de catalogar monumentos históricos na Itália. Dentre eles é elencado a modelagem baseada em imagem e a digitalização a laser por alcance. A modelagem baseada em imagem se trata de uma técnica que através de uma imagem estática, modela-se um objeto tridimensional que se assemelhe ao real e, posteriormente, mapeia-se a imagem como textura, utilizando os limites geométricos e sombras das imagens como ponto de referência para montar o relevo. Na Figura 2, pode ser observado na esquerda uma foto real do Castel Valer na Itália e a direita uma modelagem 3D do mesmo.

Figura 2– Castel Valer Real x modelagem 3D.



Fonte: Gonzo *et al.* (2007).

Embora seu custo seja reduzido em comparação com outros métodos, essa digitalização necessita que a câmera esteja nas condições perfeitas que, segundo os autores, são consideravelmente difíceis de atingir e o tempo desprendido para fazer essa modelagem é consideravelmente alto. Outro método de modelagem é a digitalização a laser por alcance, também conhecido como LiDAR. Utilizando essa forma de digitalização o processamento é rápido e capaz de detalhar ambientes complexos sem muito esforço. O seu ponto negativo é que pelo fato de mapear apenas as distancias para um objeto 3D, ele sozinho precisaria de muita edição para se mostrar útil. Logo, a sugestão levantada por Gonzo *et al.* (2007) é a de combinar os dois métodos, o fotográfico para a texturização do ambiente e o laser para a criação das modelagens, que é utilizado neste estudo.

2.2 ARKIT

ARKit se trata de um framework que contempla algumas abstrações para trabalhar com realidade aumentada nos dispositivos iOS, que mescla a utilização da visualização da câmera com os outros sensores do dispositivo. O ARKit é capaz de prover uma série de Application Programing Interfaces (APIs) para se trabalhar com realidade aumentada, dentre elas estão a Depth API, a Scene Geometry e a Instant AR, que se destacam no uso para digitalização de objetos com utilização do LiDAR, introduzidas no ARKit 4 e aprimoradas no ARKit 5. A combinação de ambas são capazes de gerar uma cena e entender cada objeto dessa cena.

Depth API, como é especificado por Apple (2022c), tem como premissa, utilizar os dados de profundidade do ambiente gerados por pixel pelo LiDAR para gerar *mashes* (polígonos 3D) que são interpretados pelo Scene Geometry, permitindo entender o que cada objeto na cena representa. Além disso, o Scene Geometry também é capaz de aplicar física do mundo real em objetos virtuais fazendo aplicações de realidade aumentada mais imersivas. Por último, é relacionado também ao Depth API ao Instant AR, que é capaz de identificar superfícies planas e ancoras na cena quase instantaneamente, fazendo com que a digitalização seja muito mais fluida. As três APIs combinadas com auxílio do framework como um todo são capazes de criar cenas complexas de realidade aumentada com oclusão de objetos virtuais

(quando um objeto virtual fica atrás de um objeto real) e com a adição do LiDAR é possível mapear cenas inteiras e de transformá-las em objetos 3D, além da capacidade de projetar objetos.

2.3 TRABALHOS CORRELATOS

Nessa seção são apresentados dois trabalhos acadêmicos e uma descrição de funcionalidade de um aplicativo que discursa sobre digitalização tridimensional, trazendo semelhanças em outras áreas acadêmicas com o presente escrito. Serão apresentados em três quadros comparativos que contribuirão para uma compreensão mais objetiva de cada trabalho correlato.

O Quadro 1 apresenta um artigo que compara os sensores de profundidade (LiDAR e TrueDepth) da Apple com soluções industriais para digitalização de objetos 3D (VOGT; RIPS; EMMELMANN, 2021). O Quadro 2 traz outro artigo que aborda uma digitalização de objetos tridimensionais utilizando o Kinect V1 e uma câmera digital de alta resolução (LOURA et al., 2018). Por fim, no Quadro 3 é evidenciada a descrição de funcionamento de um aplicativo comercial chamado Polycam, que faz a digitalização de objetos tridimensionais utilizando o LiDAR dos dispositivos Apple e disponibiliza objetos virtuais 3D (POLYCAM, 2021).

Quadro 1– Comparison of iPad Pro®'s LiDAR and TrueDepth Capabilities with an Industrial 3D Scanning Solution.

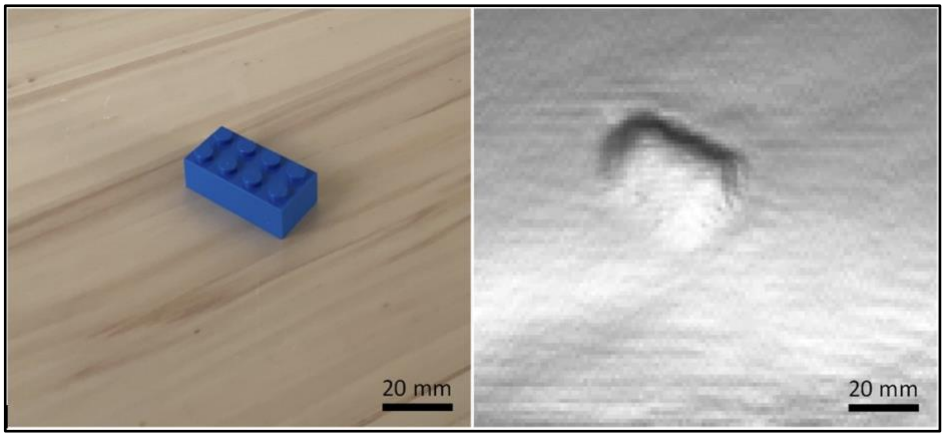
Referência	Vogt, Rips e Emmelmann (2021).
Objetivos	Avaliar a capacidade de digitalização 3D dos sensores LiDAR e True Depth se comparados com soluções industriais.
Principais funcionalidades	Digitalização 3D de objetos usando LiDAR e True Depth.
Ferramentas de desenvolvimento	Aplicativo Hedges 3D com MeshLab para tratamento do objeto digitalizado.
Resultados e conclusões	O LiDAR ainda não está pronto para o consumidor final utilizar como uma ferramenta de digitalização 3D para objetos pequenos devido a sua baixa resolução.

Fonte: elaborado pelo autor.

Nesse primeiro trabalho correlato, apresentado através do artigo desenvolvido por Vogt, Rips e Emmelmann (2021) é possível descobrir se as tecnologias dos dispositivos Apple se comparam com ferramentas comerciais já consolidadas no mercado. Sua premissa é medir a real capacidade da digitalização tridimensional com as tecnologias LiDAR e TrueDepth do iPad Pro 2020, a partir da digitalização de blocos de lego de diferentes cores e tamanhos.

Para isso, foram criados diversos cenários e observado como cada tecnologia se comporta. A tecnologia de mercado que foi contemplada é a Artec Space Spider com Blue Light Technology, que se trata de uma ferramenta muito comum para digitalização 3D de objetos (VOGT; RIPS; EMMELMANN, 2021). Do lado da Apple foi utilizado apenas o TrueDepth para digitalização, visto que o LiDAR não consegue extrair características em objetos pequenos. Na Figura 3 pode ser observado a esquerda uma peça de lego, objeto real de dimensão pequena, que quando digitalizado usando o lidar perde maior parte de suas características, como pode ser observado na imagem em tons de cinza a direita.

Figura 3 – Digitalização usando sensor LiDAR do iPad Pro 2020.



Fonte: Vogt, Rips e Emmelmann (2021).

Por fim, os autores entendem que a tecnologia ainda não está pronta para o consumidor final utilizar como uma ferramenta de digitalização 3D para objetos pequenos, devido a sua baixa resolução.

O segundo trabalho correlato estudado foi de Loura *et al.* (2018), conforme apresentado resumidamente no Quadro 2.

Quadro 2 – Reconstrução 3D de Objetos com Kinect e Câmera Digital.

Referência	Loura *et al.* (2018).
Objetivos	Realizar a digitalização 3D para preservação histórica de artigos de museu utilizando um Kinect e uma câmera digital.
Principais funcionalidades	A digitalização 3D usando o sensor Kinect e uma câmera digital.
Ferramentas de desenvolvimento	Um software próprio que utiliza Open CV, o Kinect Fusion da própria Microsoft e o driver open-source OpenKinect.
Resultados e conclusões	O resultado para os autores foi acima do esperado sendo evidenciado uma porcentagem baixa (geralmente abaixo de 10%) para erros positivos (quando a geometria do objeto virtual é maior do que do objeto real e erros negativos, quando a geometria do objeto real é maior do que a do objeto virtual).

Fonte: elaborado pelo autor.

O trabalho de Loura *et al.* (2018) relata uma solução de baixo custo que utiliza um Kinect V1, uma câmera digital e um software desenvolvido por eles para fazer a digitalização de objetos reais os transformando em objetos Computer Aided Design (CAD) que podem ser manipulados em um software de modelagem 3D posteriormente. Os objetos escolhidos pelos autores para serem digitalizados foram obras de artes consideradas culturalmente importantes do Museu de Arqueologia e Etnologia (MAE), na Universidade Federal da Bahia.

Para comparar resultados com ferramentas externas foi utilizado um programa chamado Kinect Fusion que também faz a digitalização de objetos. Dado os resultados encontrados foi perceptível que a solução dos escritores se mostrou precisa o suficiente para fazer uma digitalização de qualidade deixando apenas alguns ruídos na digitalização. Uma comparação entre os objetos virtuais (colunas ímpares com o fundo azul) e objetos reais podem ser observados na Figura 4.

Figura 4 – Objetos virtuais e objetos reais.

Fonte: Loura *et al.* (2018).

Por fim, apresenta-se no Quadro 3, uma descrição de funcionalidade de um aplicativo como trabalho correlato do presente estudo.

Quadro 3– Polycam.

Referência	Polycam (2021).
Objetivos	Aplicativo que gera modelos 3D a partir de escaneamento do ambiente utilizando o LiDAR e permite a exportação como um Computer Aided Design (CAD).
Principais funcionalidades	Escaneamento 3D, compartilhamento de objeto e exportação.
Ferramentas de desenvolvimento	Swift.

Fonte: elaborado pelo autor.

Polycam (2021) é um aplicativo de código fechado para os dispositivos móveis da Apple que permite a digitalização de um ambiente e sua manipulação, recortando áreas e isolando objetos. A digitalização é feita a partir de um mapeamento de polígonos utilizando o sensor LiDAR que consegue determinar a profundidade do ambiente. Já a texturização é feita através de uma filmagem usando a câmera do dispositivo, que com as imagens capturadas se torna possível criar a textura do objeto tridimensional virtual. Na Figura 5 é possível observar a tela do aplicativo durante uma

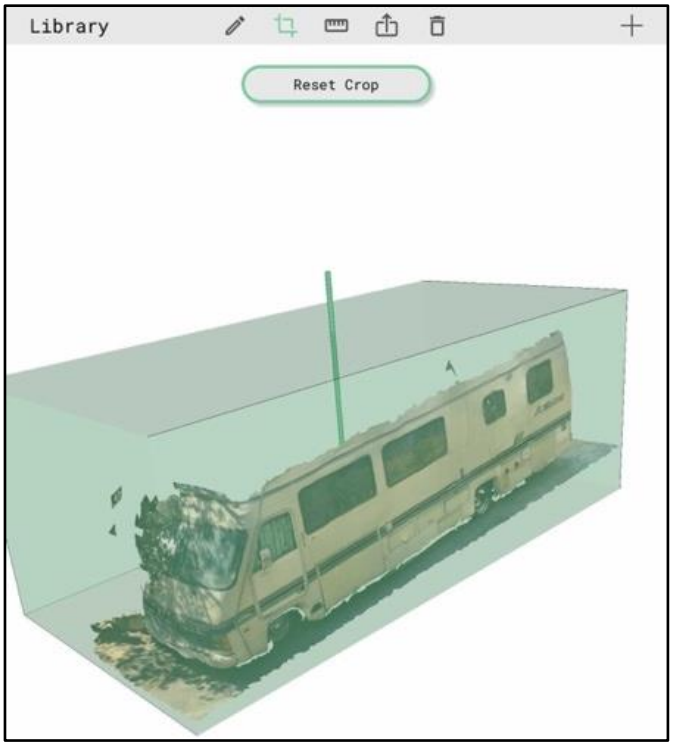
digitalização onde as partes destacadas em azul representam áreas não mapeados e os polígonos com linhas brancas representam a área mapeada.

Figura 5 – Digitalização de um ambiente sendo feito utilizando o Polycam.

Fonte: Polycam (2021).

Após a finalização da digitalização, do mapeamento do objeto e sua textura, é possível delimitar a área do ambiente capturado para fazer um recorte apenas do objeto de interesse, como pode ser observado na Figura 6.

Figura 6 – Interface de edição de objeto do Polycam.



Fonte: Polycam (2021).

Conforme imagem, pode-se compreender que é possível delimitar uma área que formará o objeto final utilizando a ferramenta de corte que é representada pelo paralelepípedo verde na imagem.

3 DESCRIÇÃO DA BIBLIOTECA

Nessa seção serão descritos as funcionalidades e o desenvolvimento da biblioteca do ARKit. Com essa finalidade, foram criadas quatro subseções. Na primeira subseção, apresenta-se a especificação da biblioteca com os diagramas de classe e fluxo a tela do aplicativo. Na segunda subseção, descreve-se a implementação destacando os principais trechos

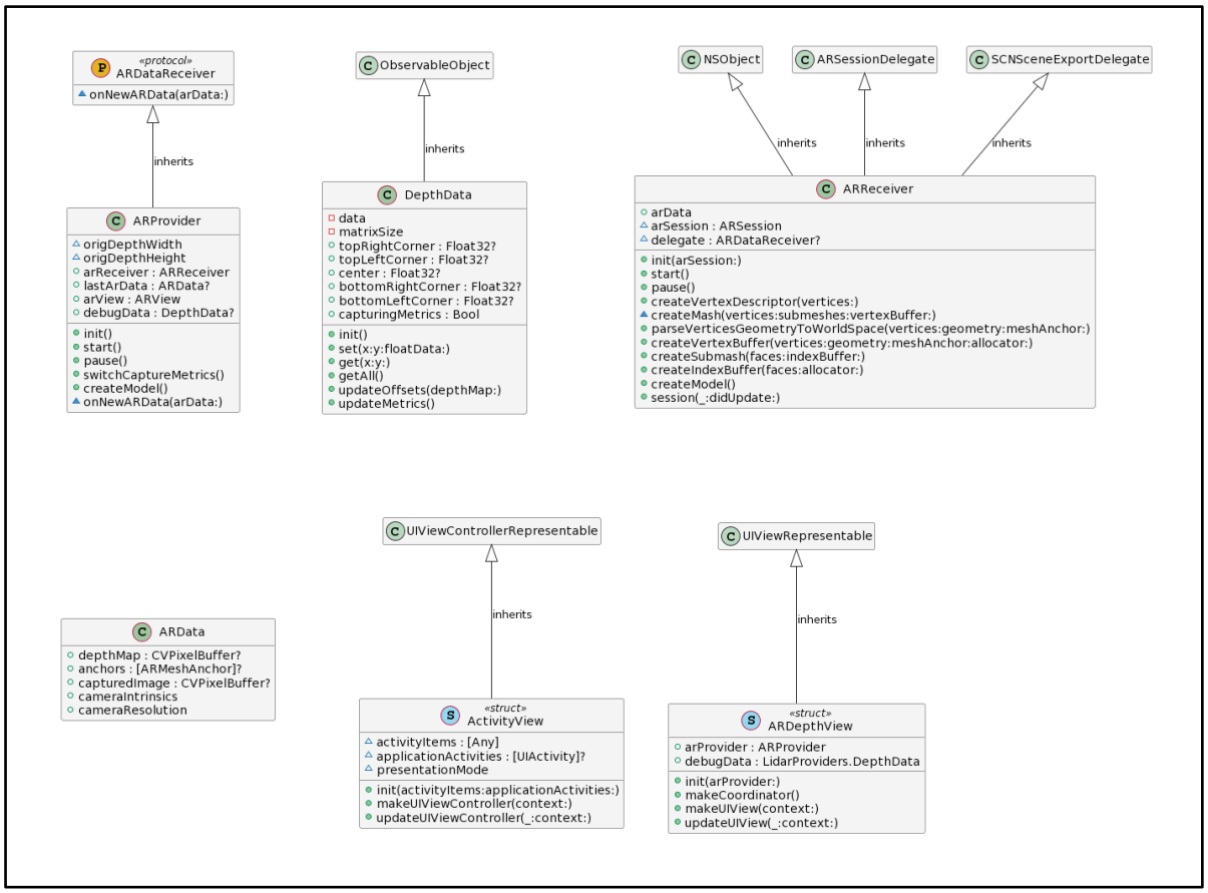
de código da biblioteca. Na terceira subseção, demonstra-se a instalação da biblioteca em um aplicativo iOS e sua utilização. Por último, na quarta seção, serão mostrados os cenários de testes realizados.

3.1 ESPECIFICAÇÃO

Para desenvolver a biblioteca foi utilizada a linguagem de programação Swift na sua versão 5.7 e o encapsulamento em uma biblioteca foi feito através de um Swift Package, que pode ser adicionado em qualquer projeto Swift. Sua documentação com alguns detalhes extras pode ser encontrada no Apêndice A – Documentação da biblioteca. Para fazer o mapeamento do ambiente com o LiDAR foi utilizada a Depth API do *framework* ARKit, em conjunto com a `SceneGeometry` do *framework* SceneKit onde são capturadas as informações dos pontos no mundo e salvos em um objeto de *meshes* que, posteriormente, com o objeto construído, serão salvos em um arquivo `.dae` para serem manipulados em um software de modelagem 3D.

A biblioteca possui sete classes onde duas delas são *providers* (classes que contém métodos que fornecem funcionalidades para serem utilizadas por outras classes), o `ARReceiver` e o `ARDataProvider`, outras duas são *dataclasses* (classes que apenas carregam dados), a `ARData` e a `DepthData`, duas são *views* (classes que geram componentes para serem exibidos em tela), a `ActivityView` e a `ARDepthView`, por último, um *protocol* (classe que define os métodos que quem a estende deve implementar), a `ARDataReceiver`. Na Figura 7 é possível visualizar o diagrama de classes da biblioteca.

Figura 7 – Diagrama de classes da biblioteca.

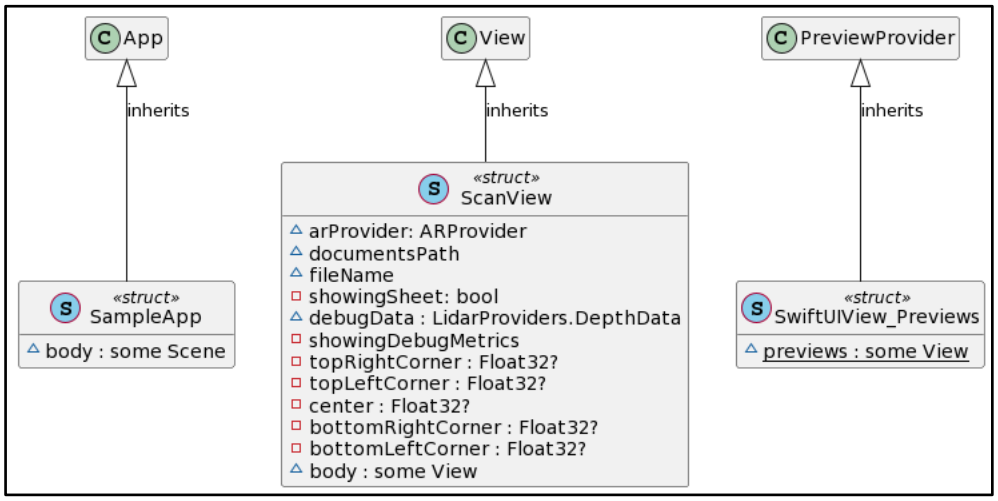


Fonte: elaborado pelo autor.

Também foi criada uma aplicação que utiliza a biblioteca a fim de testar e comprovar as funcionalidades. Essa aplicação funciona em qualquer dispositivo iOS que possua o LiDAR e é formada pela `ScanView`, que utiliza SwiftUI com um atributo de `ARDepthView`, disponibilizada na visualização da câmera. Essa aplicação também possui um botão que apresenta métricas de *debug* através da *dataclass* `DepthData` e um botão que salva um modelo 3D no dispositivo chamando o método `createModel()` do objeto `arProvider` e permite compartilhar esse modelo utilizando o modal de compartilhamento do iOS provido pela *view* `ActivityView`.

Na Figura 8, é possível observar o diagrama de classes dessa aplicação onde além da `ScanView`, descrita acima, também consta o `SampleApp` (representa a *view* principal, chamada na execução da aplicação e que dentro de seu `body` possui uma instancia de `ScanView`), bem como a `SwiftUIView_Preview` (faz o filtro pelo tipo de dispositivo e, caso o LiDAR não esteja disponível no dispositivo, ele exibe uma mensagem de erro).

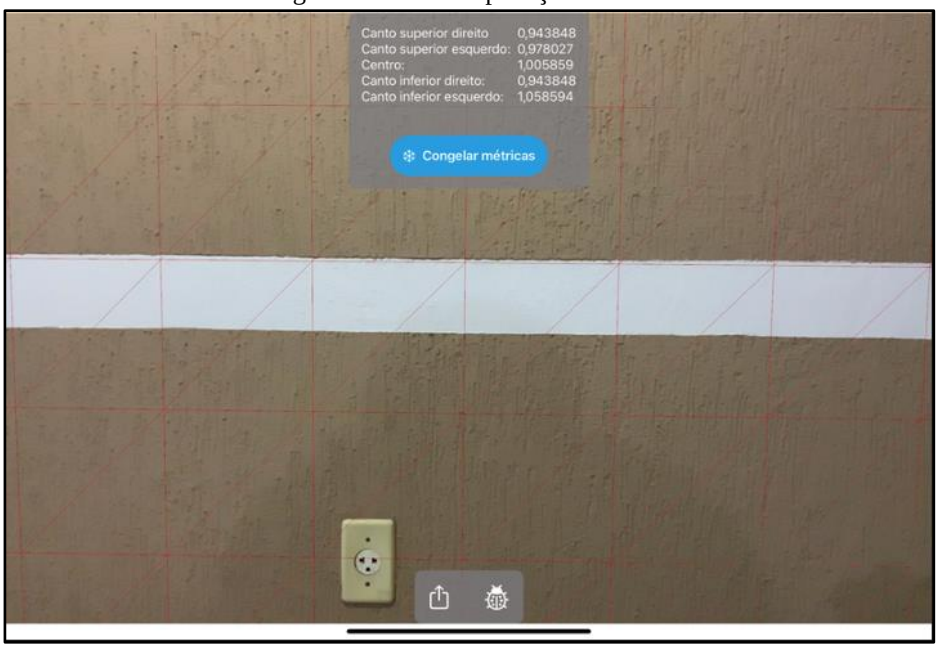
Figura 8 – Diagrama de classes da aplicação.



Fonte: elaborado pelo autor.

A interface dessa aplicação foi criada de forma a contribuir com praticidade ao se realizar os testes necessários, possibilitando a exportação de objetos 3D. Na Figura 9, é possível observar que a tela principal possui dois botões na parte inferior, o esquerdo serve para compartilhar o modelo mapeado até o momento e o direito serve para abrir o modal que se encontra na parte superior da tela, demonstrando as distâncias nos *pixels* de cada canto da tela e do centro da tela para motivos de testes.

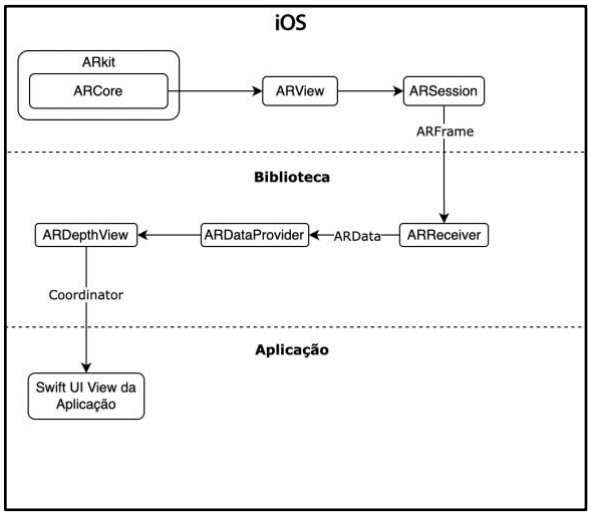
Figura 9 – Tela da aplicação de teste.



Fonte: elaborado pelo autor.

Para facilitar a implementação mais comum de uma aplicação de escaneamento a classe `ARDepthView` é disponibilizada, no qual se for adicionada ao `Body` de uma `SwiftUI`, ela cria um objeto de `ARDataProvider` que disponibiliza as APIs necessárias para o escaneamento 3D e demonstra uma `ARView`. Na Figura 10, pode-se observar um diagrama com o passo a passo feito pelo fluxo descrito acima.

Figura 10 – Diagrama de Fluxo do funcionamento da biblioteca.



Fonte: elaborado pelo autor.

Com o contexto exercitado nessa seção de uma forma mais genérica, vale ressaltar que esse fluxo não é a única forma de utilizar a biblioteca, pois é possível utilizar cada *provider* independentemente de uma *view*.

3.2 IMPLEMENTAÇÃO

A seção de implementação foi segregada em três módulos principais. O primeiro modulo foi idealizado para categorizar melhor o `ARReceiver` o segundo categoriza o ARProvider e o último elenca as principais funcionalidades do `ARDepthView`.

3.2.1 ARReceiver

A cada atualização do `ARSession`, um fluxo é gerado e um objeto de `ARFrame` com as atualizações capturadas pela câmera e pelos sensores do dispositivo é enviado para todas as classes que estendam o protocolo `ARSessionDelegate` e sobrescrevam o método `session(_didUpdate)`. Para a biblioteca, essa classe é o `ARReceiver`, em que a cada frame de atualização captura as informações necessárias para a digitalização 3D e guarda em um objeto de `ARData`.

Porém, caso quem tenha instanciado o `ARReceiver` passe como parâmetro para a variável `delegate` uma classe que estenda o *protocol* `ARDataReceiver`, será chamado o método `onNewARData(arData: ARData)` desse `delegate`, para cada chamada do método `session(_didUpdate)`. No Quadro 4 é possível observar a implementação do `session(_didUpdate)`.

Quadro 4 – Código do método `session(_didUpdate)` no `ARReceiver`.

```
public func session(session: ARSession, didUpdate frame: ARFrame) {
    if(frame.sceneDepth != nil) {
        arData.depthMap = frame.sceneDepth!.depthMap
        arData.anchors = frame.anchors.compactMap({ $0 as? ARMeshAnchor })
        arData.capturedImage = frame.capturedImage
        arData.cameraIntrinsics = frame.camera.intrinsics
        arData.cameraResolution = frame.camera.imageResolution
        delegate?.onNewARData(arData: arData)
    }
}
```

Fonte: elaborado pelo autor.

Para realizar a escaneamento o `ARReceiver` dispõe do método `createModel()` que é responsável por gerar o modelo 3D, utilizando os dados contidos no `arData` atualizado a cada chamada do `session(_didUpdate)`, conforme pode ser observado no Quadro 5.

Quadro 5 – Código que gera o modelo 3D.

```
public func createModel() -> SCNScene {
    // Nessa etapa é criado um alocador de memória, para guardar os
    // dados do objeto 3D até ser persistido
    let allocator = MTKMeshBufferAllocator(device:
EnvironmentVariables.shared.metalDevice)
    // Com o alocador criado, um asset que irá montar o objeto 3D é criado
    let asset = MDLAsset(bufferAllocator: allocator)
    // aqui é feita uma verificação para garantir que
    // a instancia de ARData está preenchida com as ancoras do ambiente
    guard let anchors = self.arData.anchors else { fatalError("No anchors were
found") }
    // Então uma iteração é feita pelas ancoras encontradas
    for meshAnchor in anchors {
        // Recupera as geometrias da cena AR
        let geometry = meshAnchor.geometry
        let faces = geometry.faces

        // Cria um buffer de índices que serão utilizadas como
        // a geometria do objeto 3D.
        let indexBuffer = createIndexBuffer(faces: faces, allocator: allocator)

        // e adiciona em uma sub mash
        let submesh = createSubmash(faces: faces, indexBuffer: indexBuffer)

        // Cria os vértices que conectam as faces e a partir deles
        // cria a mash principal que será adicionada ao modelo 3D
        let vertices = geometry.vertices
        let vertexBuffer = createVertexBuffer(vertices: vertices, geometry:
geometry, meshAnchor: meshAnchor, allocator: allocator)
        let mesh = createMash(vertices: vertices, submeshes: [submesh],
vertexBuffer: vertexBuffer)

        // Adiciona a mash gerada ao modelo 3D
        asset.add(mesh)
    }
    return SCNScene(mdlAsset: asset)
}
```

Fonte: elaborado pelo autor.

A classe `ARReceiver` ainda dispõe de dois métodos para inicializar ou congelar a detecção e métodos privados que modularizam a criação do modelo 3D, porém, todos eles são chamados no `createModel()`.

3.2.2 ARProvider

A classe `ARProvider` é uma extensão do *protocol* `ARDataProvider` e em seu construtor, um objeto de `ARReceiver` é criado sendo que seu `delegate` é preenchido com `self` (a instancia da própria classe `ARProvider`). Logo, a cada atualização do `ARSession` o método `onNewARData(arData: ARData)` do `ARProvider` é chamado com a atualização do `arData`.

No Quadro 6, pode-se verificar a implementação do `onNewARData(arData: ARData)` na classe `ARProvider`. Nesse método, é possível observar a atualização de uma variável local chamada `lastArData` com o `arData` recebido pelo `ARReceiver`.

Quadro 6 - `onNewARData(arData: ARData)` da classe `ARProvider`.

```
func onNewARData(arData: ARData) {
    lastArData = arData
    #if DEBUG
    if !(debugData?.capturingMetrics ?? false) {
        return
    }
    if let depthMap = arData.depthMap {
        DispatchQueue.main.async {
            self.debugData?.updateOffsets(depthMap: depthMap)
            self.debugData?.updateMetrics()
        }
    }
    #endif
}
```

Fonte: elaborado pelo autor.

Caso a aplicação esteja sendo rodada no modo de depuração (*debug*) um objeto de `DepthData` é salvo na variável `debugData` com as profundidades de cada pixel detectado pelo LiDAR. Vale ressaltar que a matriz de pixels detectados pelo LiDAR é de 256 de largura por 192 de altura.

3.2.3 ARDepthView

Para a utilização do `ARProvider` em uma *view* é necessário que seja criada uma `ARView` e alguns parâmetros sejam configurados. No Quadro 7 é possível constatar que são configuradas algumas opções, e cada comentário explica o que cada linha significa.

Quadro 7 - Método de configuração da `ARDepthView`.

```
public func makeUIView(context: UIViewRepresentableContext<ARDepthView>) -> ARView
{
    let arView = arProvider.arView

    // O coordinator do parâmetro context carrega uma instância de ARReceiver,
    // que é atribuído para a session do ARView, logo, cada atualização no
    // ARSession será repassada para o ARReceiver.
    arView.session.delegate = context.coordinator

    arView.environment.sceneUnderstanding.options = []
    // Liga a oclusão da cena para a reconstrução da mesh.
    arView.environment.sceneUnderstanding.options.insert(.occlusion)
    // Liga a oclusão da cena para a reconstrução da mesh.
    arView.environment.sceneUnderstanding.options.insert(.physics)
    // [OPCIONAL] demonstra o mapeamento da mesh na visualização
    arView.debugOptions.insert(.showSceneUnderstanding)
    // Desabilita as opções pois não são necessárias
    arView.renderOptions = [.disablePersonOcclusion, .disableDepthOfField,
.disableMotionBlur]
    // Desabilita a configuração automática da sessão
    arView.automaticallyConfigureSession = false

    return arView
}
```

Fonte: elaborado pelo autor.

Por fim, com todas as classes descritas, é possível compreender a estrutura de cada componente da biblioteca e o que é necessário para implementar ela.

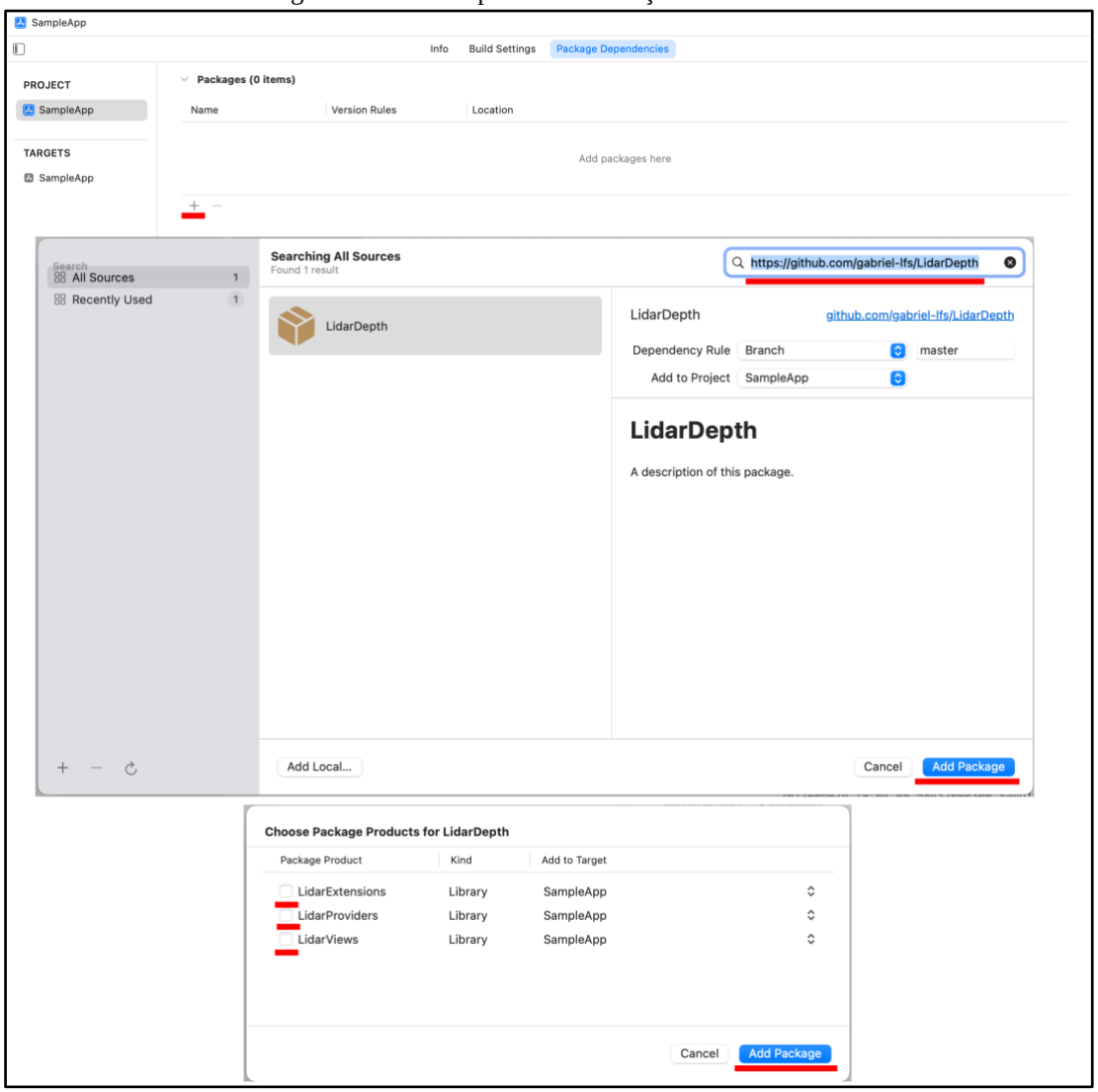
3.3 UTILIZAÇÃO E APLICAÇÃO TESTE

Nessa subseção será demonstrado a utilização da biblioteca em uma aplicação iOS. Para isso, primeiramente, é necessário fazer a instalação adicionando a biblioteca como um *remote package* através do gerenciador de pacotes do XCode. Para fazer essa adição, deve-se entrar nas configurações do projeto, clicando no nome do projeto na barra esquerda do XCode, selecionar *Package Dependencies* e acessar ao sinal de mais, embaixo da tabela de pacotes. Assim, uma janela irá abrir, no qual deve ser escrito no campo de busca a URL do projeto no GitHub `https://github.com/gabriel-lfs/LidarDepth` e clicar em Add Package.

Na próxima janela aberta, pode-se escolher quais módulos da biblioteca se deseja instalar, onde `LidarExtensions`, é um pacote com extensões das classes para implementações relacionadas a modelagem 3D, esse

pacote é requerido para instalar os outros dois pacotes, porém caso desejado, ele pode ser instalado sozinho. O outro pacote é o `LidarProviders` que depende do `LidarExtensions`, onde encontram-se as classes `ARProvider`, `ARDataProvder`, `ARReceiver` e `DepthData` mencionadas nas seções anteriores. Por último, também está disponível o pacote `LidarViews` que depende dos outros dois pacotes para seu funcionamento e disponibiliza as duas *views* `ARDepthView` e `ActivityView`. Na Figura 11, pode-se observar as telas do XCode com o passo a passo para fazer a adição.

Figura 11 - Passo a passo da instalação da biblioteca.



Fonte: elaborado pelo autor.

Com a instalação efetivada, para utilizar a biblioteca, basta importar o pacote desejado (`LidarExtensions`, `LidarProviders`, `LidarViews`) e chamar a classe desejada. No Quadro 8, é possível verificar uma implementação básica que utiliza o `ARProvider` para ler os dados do AR e quando um botão é clicado, chama-se o método que faz o escaneamento 3D.

Quadro 8 - Implementação básica do `ARProvider`.

```
import SwiftUI
import LidarProviders

struct ScanView: View {
    let arProvider = LidarProviders.ARProvider()

    var body: some View {
        Button(action: {
            let asset = self.arProvider.createModel()
            // a partir daqui a constante asset carrega o modelo 3D gerado
        }) {
            Image(systemName: "square.and.arrow.up")
                .font(.title)
                .foregroundColor(.white)
        }
    }
```

Fonte: elaborado pelo autor.

Qualquer método pode ser chamado a partir do `arProvider`. Porém por seu uso sempre conter uma *view* envolvida é necessário criar um *coordinator* e configurar a *view* da forma que foi descrita na sessão 3.2.3 para demonstrar como um painel de visualização com o `SwiftUI` . Assim, a exibição do `ARView` terá os atributos necessários para fazer o escaneamento 3D. A forma mais simples de criar esse *coordinator* é instanciando um `ARDepthView` que é importado de `LidarViews` e adicionando essa instancia no `body` da *view* do `SwiftUI`.

Com todo o contexto exercitado neste e nos tópicos anteriores, foi criada uma aplicação que utiliza a biblioteca, gerando um modelo 3D que disponibiliza métricas de profundidade. Essa aplicação tem como intuito validar a viabilidade de uso da biblioteca no contexto de digitalização 3D, além de testar a precisão do sensor LiDAR na questão de distância. Para isso, as métricas são calculadas a cada nova atualização de AR e um objeto de `DepthData` é atualizado com a matriz de profundidade de todos os pixels capturados pelo LiDAR, permitindo assim que seja gerada uma visualização.

3.4 CENÁRIOS DE TESTE

A aplicação foi criada com a intenção de descobrir a real funcionalidade do sensor LiDAR através das implementações da biblioteca. Para isso, foram criados cenários de testes que testam os limites desse sensor.

Segundo Apple (2022a), o LiDAR é capaz de medir até 5 metros de profundidade, então foram criados alguns cenários de teste em um ambiente controlado de 6,91 metros que foi demarcado com duas trenas de três metros e uma de um metro e meio que mediu os últimos noventa e um centímetros. O dispositivo utilizado foi um iPad Pro 2020, posicionado em diferentes distâncias com uma variância de luz entre 44 lux e 0 lux (com uma luz incandescente acesa e com nenhuma luz acesa respectivamente). Na

Figura 12, pode ser observado o cenário selecionado para os testes, onde o dispositivo era utilizado de maneira móvel para mensuração, enquanto o objeto estático era a parede.

Figura 12 - Ambiente utilizado para testes.

Fonte: elaborado pelo autor.

Ainda, para testar a capacidade de medida foi utilizado um objeto de vinte e cinco centímetros como obstáculo para a medida, com intuito de validar se a proporção é mantida mesmo nos erros encontrados, conforme pode ser observado na Figura 13.

Figura 13 - Objeto utilizado como obstáculo.

Fonte: elaborado pelo autor.

Por fim, com todo o levantamento, os testes foram realizados em três distâncias diferentes, com a luminância de 44 lux e de 0 lux. As distâncias foram, de 1 metro de distância (com a intenção de testar o cenário mais confortável para o sensor), de 4 metros (testando o limite do sensor) e de 6,91 metros (superando o limite do sensor). Nessas distâncias foram medidos os cantos superiores esquerdo e direito e o centro da tela. Os cantos inferiores não serão contemplados, pois em maior parte dos testes estavam medindo a distância do chão. No ApênDice B – Capturas de tela dos Testes podem ser encontradas as capturas de telas que evidenciam cada um dos testes

4 RESULTADOS

Para esse estudo, foi decidido testar as capacidades do LiDAR em diferentes cenários. Para isso, duas baterias de testes foram reproduzidas. A primeira são medições de distância e a segunda são escaneamentos 3D. Nessa seção serão demonstrados os resultados para cada um dos testes efetuados e discursada a semelhança encontrada nos trabalhos correlatos.

4.1 MEDIÇÕES DE DISTÂNCIA

Nessa subseção irá ser exercitado os testes descritos no tópico 3.4. Para isso, serão elencadas quatro tabelas com as distancias e a condição de iluminação variantes. Na Tabela 1 serão demonstradas as distancias de 1 metro, 4 metros e 6,91 metros considerando o ponto central da matriz de profundidade gerada pela `DepthAPI`. Vale salientar que ocorre uma flutuação das casas decimais constante, mesmo com o dispositivo e o alvo da medição estatícos, e que essa flutuação aumenta conforme a distância do alvo aumenta.

Ainda na Tabela 1, pode-se observar que a precisão das medidas vai diminuindo conforme o sensor se afasta do alvo da medição. Porém, dentro da distância contemplada de 5 metros previamente elencada, o sensor consegue uma precisão de até duas casas decimais e que a partir da distância não contemplada, a flutuação começa a ser na primeira casa decimal.

Tabela 1- Medição da distância no ponto central detectado pelo sensor.

Distância real	Distância detectada com luz acesa	Distância detectada com luz apagada
1 metro	1,005859 metro	1,004883 metro
4 metros	4,007812 metros	4,007812 metros
6,91 metros	6,371094 metros	7,445312 metros

Fonte: elaborado pelo autor.

Na Tabela 2, pode ser observado os mesmos testes, porém pela perspectiva do ponto superior direito capturado pelo sensor. Também é possível observar que a medida de 6,91 não contém valores, visto que nessa distância não foi

possível obter uma medição dado o cenário de teste, pois os pontos superiores estavam medindo a distância até o teto e os inferiores mediam a distância até o chão. Isso não se mostrou um problema, já que no ponto central, tinha sua flutuação grande demais para ser desconsiderada, diferentemente de testes anteriores onde o ponto central, até a segunda casa decimal, era preciso.

Tabela 2 - Medição da distância no ponto lateral superior direito detectado pelo sensor.

Distância real	Distância detectada com luz acesa	Distância detectada com luz apagada
1 metro	0,943848 metro	0,968262 metro
4 metros	3,818359 metros	3,951172 metros
6,91 metros	-	-

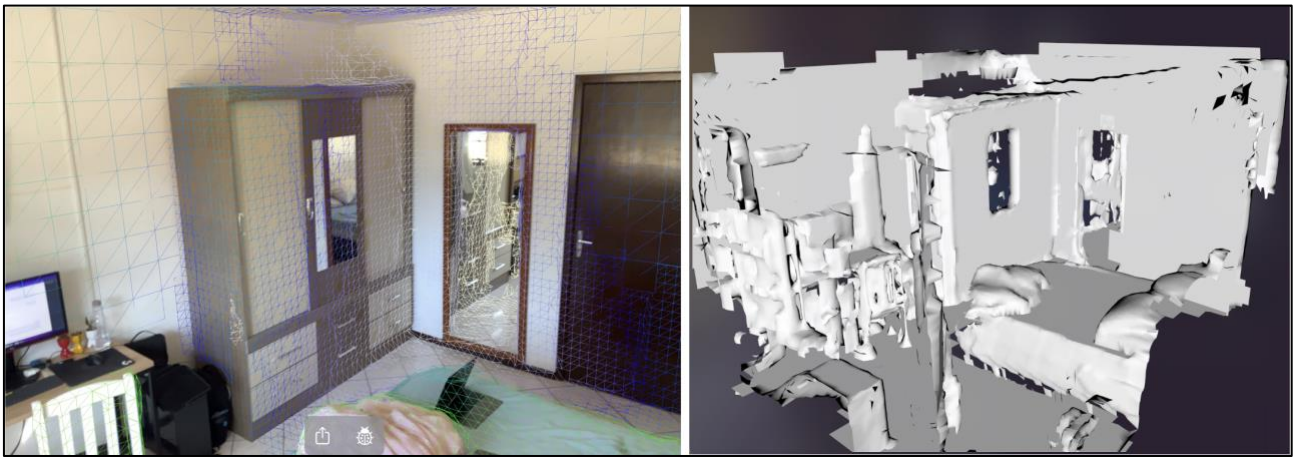
Fonte: elaborado pelo autor.

A partir da análise da Tabela 1 e a Tabela 2, pode-se observar que, para um mesmo objeto, todos os pontos que são capturados nos cantos detectam mais proximidade do que o ponto central, independente da distância. Pode-se observar também que dentro da distância prevista pela Apple, o ponto central é preciso até a segunda casa decimal. Logo, é possível concluir que a detecção é de forma concava com erro de aproximadamente 5 até 20 centímetros a menos de distância nas extremidades do sensor. Também pode ser notado que com a luz apagada a detecção se torna mais precisa para os cantos, ficando com uma diferença de 2 a 5 centímetros. O que pode ser ocasionado por conta dos sensores que são utilizados para fazer a medição, sem uma fonte de luz, a câmera não pode ser utilizada para gerar uma estimativa, fazendo com que o LiDAR precise trabalhar sozinho.

4.2 DIGITALIZAÇÃO 3D

Nessa subseção irá ser abordado a capacidade de fazer o escaneamento 3D utilizando o LiDAR. Assim como no trabalho de Vogt, Rips e Emmelmann (2021), o LiDAR demonstrou dificuldade para fazer o escaneamento 3D utilizando a biblioteca desenvolvida no presente escrito, perdendo características dos objetos e gerando uma malha com baixa quantidade de polígonos. O principal ponto positivo é a sua velocidade de digitalização, pois o mapeamento dos objetos é feito em segundos e a facilidade de uso, já que o objeto gerado de `SCNScene` é manipulável a nível de aplicação, podendo ser utilizado em algoritmos de processamento de imagem ou externamente com suas possíveis exportações em `.dae` , para manipulação em macOS, ou em `.scn`, para manipulação com contexto de realidade aumentada como discrimina Apple (2022d). Na Figura 14, é possível observar um quarto em sua forma real (em colorido) e a digitalização 3D desse mesmo quarto (em cinza, branco e preto).

Figura 14 - Escaneamento 3D de um quarto.



Fonte: elaborado pelo autor.

A partir desse escaneamento é possível notar que muitas características são de fato perdidas, mas a forma geral dos objetos é mantida. Outra observação percebida é que o sensor, quando apontado para um espelho, faz o mapeamento do ambiente como se os objetos espelhados fossem sólidos e tivessem uma forma, representando o próprio espelho como um buraco na parede.

O que pode se entendido, se comparado com o estudo de Loura *et al.* (2018), é que o LiDAR tem uma maturidade muito menor para fazer digitalizações 3D se comparado com o Kinect, já que seus polígonos ainda não são polidos como os gerados pelas nuvens de pontos do Kinect, e que se comparado com o sensor True Depth do FaceID no iPad, como foi testado no trabalho de Vogt, Rips e Emmelmann (2021), o LiDAR ainda assim perde em precisão de polígonos.

5 CONCLUSÕES

Por fim, o que pode ser concluído com esse trabalho, é que seus principais pontos de estudo foram alcançados, porém algumas limitações foram encontradas. O LiDAR do iPad Pro não é a solução ideal para digitalização 3D em um nível empresarial, se comparado com um sensor industrial, como foi feito pelo artigo de Vogt, Rips e Emmelmann (2021). Mas, como uma solução pratica e de maior acessibilidade (dado que está em um dispositivo móvel), o sensor demonstra ser eficiente, permitindo o mapeamento de uma superfície de forma rápida mesmo que não milimetricamente precisa. Na ótica da biblioteca, seu papel de viabilizar a digitalização 3D em um dispositivo iOS foi cumprida além disso, foi capaz de entregar facilitadores para recuperar a medição de distância do sensor LiDAR e uma interface para compartilhar arquivos, no caso, os arquivos digitalizados.

Uma limitação encontrada foi na texturização, já que, a classe `SCNScene` utilizada no objeto 3D não disponibiliza uma API para texturização, sendo necessário fazer uma sobrescrita na forma que ele cria o objeto, além de adicionar essa textura manualmente. Em uma evolução dessa biblioteca, poderiam ser utilizadas informações da imagem da câmera disponibilizadas pelo objeto de `ARData` para fazer essa texturização. Uma outra evolução possível, é relativa aos testes desempenhados. Foram testados cenários de distâncias variadas baseados nos limites de detecção do iPad além de uma variação na luz. Porém, em um futuro estudo, ainda pode ser testado movimentação do dispositivo em relação a um objeto estático, a movimentação de um objeto estático em relação ao dispositivo parado e diferentes tipos de superfície (com reflexo e com translucidez), para testar a real capacidade do sensor nas mais adversas situações. Assim, almeja-se ter contribuído, a partir do presente estudo, para a construção de uma biblioteca para digitalização de objetos 3D que reflita na construção de novas aplicações e estudos referentes a temática.

REFERÊNCIAS

APPLE. **Apple anuncia novo iPad Pro com inovador scanner LiDAR e trackpad para uso com iPadOS**, 2022a. Disponível em: https://www.apple.com/br/newsroom/2020/03/apple-unveils-new-ipad-pro-with-lidar-scanner-and-trackpad-support-in-ipados/. Acesso em: 12 Jun. 2022.

APPLE. **Framework ARKit,** 2022b. Disponível em: https://developer.apple.com/documentation/arkit/. Acesso em: 12 Jun. 2022.

APPLE. **More to Explore with ARKit 6**, 2022c. Disponível em: https://developer.apple.com/augmented-reality/arkit/. Acesso em: 12 Jun. 2022.

APPLE. **write(to:options:delegate:progressHandler:)**, 2022d. Disponível em: https://developer.apple.com/documentation/scenekit/scnscene/1523577-write. Acesso em: 12 Jun. 2022.

BAUWENS, Sébastien; BARTHOLOMEUS, Harm; CALDERS, Kim; LEJEUNE, Philippe. Forest Inventory with Terrestrial LiDAR: a comparison of static and hand-held mobile laser scanning. **Forests**, Basel, v. 7, n. 12, p. 127, 21 jun. 2016. MDPI AG. Disponível em: http://dx.doi.org/10.3390/f7060127. Acesso em: 1 Jun. 2021.

GIONGO, Marcos; KOEHLER, Henrique Soares; MACHADO, Sebastião do Amaral; KIRCHNER, Flavio Felipe; MARCHETTI, Marco. LiDAR: princípios e aplicações florestais. **Pesquisa Florestal Brasileira**, [S.L.], v. 30, n. 63, p. 231-244, 28 out. 2010. Embrapa Florestas. http://dx.doi.org/10.4336/2010.pfb.30.63.231. Acesso em: 20 set 2021.

GONZO, L.; VOLTOLINI, F.; GIRARDI, S.; RIZZI, A.; REMONDINO, F.; EL-HAKIM, S.F. Multiple Techniques Approach to the 3D Virtual Reconstruction of Cultural Heritage. *In*: EUROGRAPHICS ITALIAN CHAPTER CONFERENCE, 2, 2007, Trento. **Anais** R. De Amicis and G. Conti. p. 213 – 216. Disponível em: http://dx.doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2007/213-216. Acesso em: 21 set 2021.

LOURA, Daniel de Sousa Alves; OLIVEIRA, Yuri de Matos Alves de; RAIMUNDO, Pedro Oliveira; AGÜERO, Karl Philips Apaza. Reconstrução 3D de Objetos com Kinect e Câmera Digital. **Revista Eletrônica de Iniciação Científica em Computação**, Bahia, v. 16, n. 6, p. 1-17, 8 dez. 2018. Sociedade Brasileira de Computação - SB. Disponível em: http://dx.doi.org/10.5753/reic.2018.1077. Acesso em: 12 out 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. Acesso em: 12 out 2021.

NASSIF, Felipe de Barros; PASSOS, Júlio César; PIMENTA, Felipe Mendonça. **A Tecnologia Lidar Aplicada A Medições Eólicas Sobre Corpos Hídricos E Oceano.** 2017. 111 f. Dissertação (Mestrado) - Curso de Engenharia Mecânica, Universidade Federal de Santa Catarina, Florianópolis, 2017. Disponível em: https://repositorio.ufsc.br/handle/123456789/186188. Acesso em: 10 nov. 2021.

POLYCAM. **FAQ**. 2021 Disponível em: https://poly.cam/learn. Acesso em: 20 set 2021.

RIGUES, Rafael. **Snapchat será um dos primeiros apps a usar o Lidar no iPhone 12 Pro**. 2020. Disponível em: https://olhardigital.com.br/2020/10/14/noticias/snapchat-sera-um-dos-primeiros-apps-a-usar-o-lidar-no-iphone-12-pro/. Acesso em: 20 set. 2021.

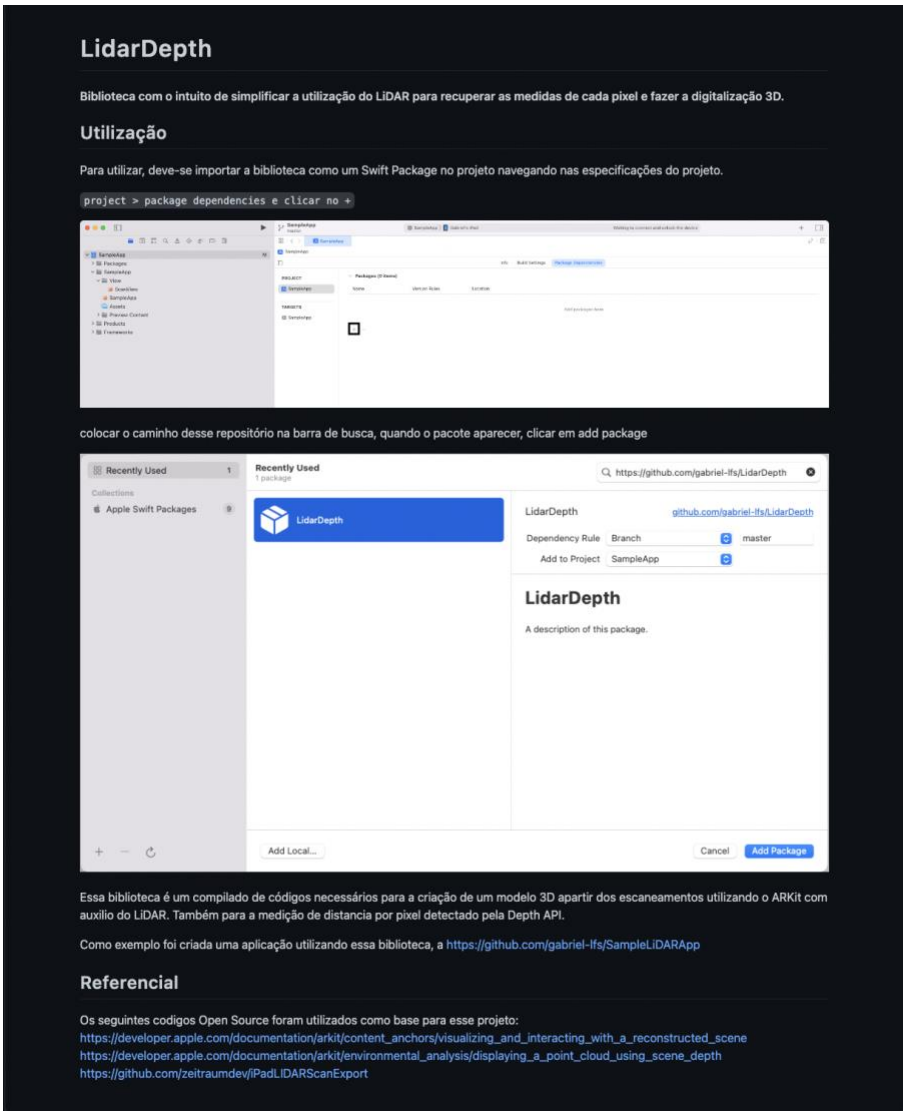
SOUZA, Gabriel. **LidarDepth**. 2022 Disponível em: https://github.com/gabriel-lfs/LidarDepth/blob/master/README.md. Acesso em: 29 jun. 2022.

VOGT, Maximilian; RIPS, Adrian; EMMELMANN, Claus. Comparison of iPad Pro®'s LiDAR and TrueDepth Capabilities with an Industrial 3D Scanning Solution. **Technologies**, Basel, v. 9, n. 2, p. 25, 7 abr. 2021. MDPI AG. Disponível em: http://dx.doi.org/10.3390/technologies9020025. Acesso em: 20 set 2021.

6 APÊNDICE A – DOCUMENTAÇÃO DA BIBLIOTECA

Nesta seção de apêndice será apresentado a documentação da biblioteca que está no GitHub, repositório de códigos. Na Figura 15 pode ser observada essa a documentação com o referencial dos códigos utilizados como base para o desenvolvimento.

Figura 15 - README.md da biblioteca.

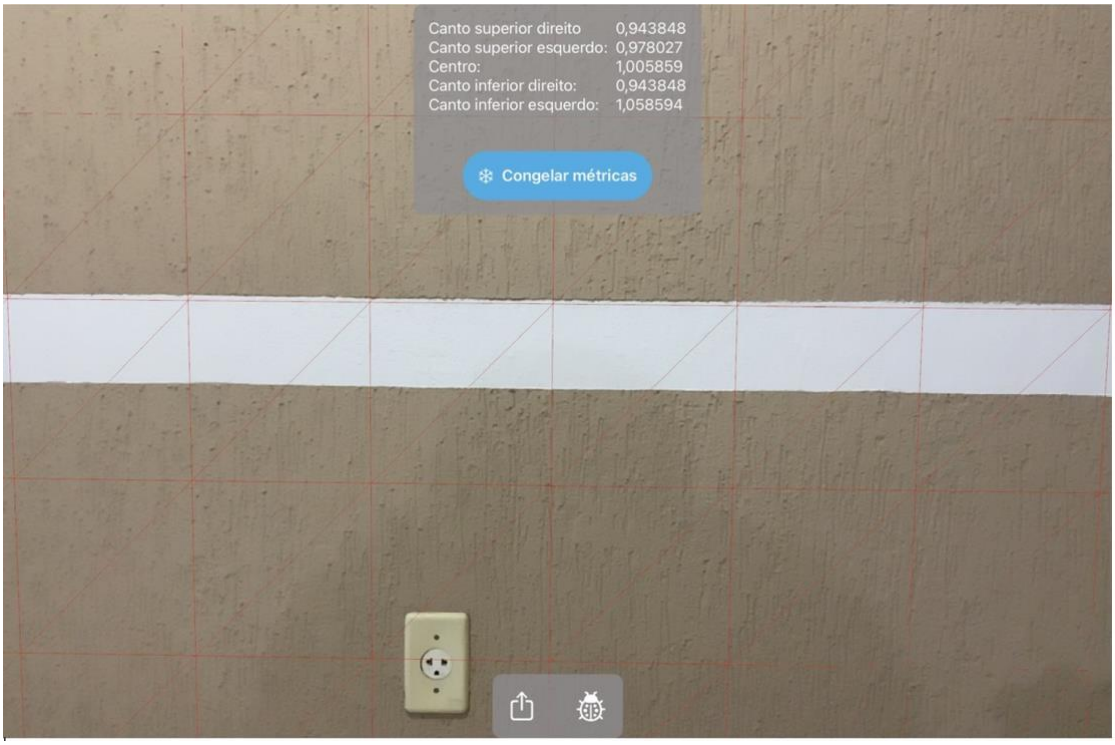


Fonte: Souza (2022).

7 APÊNDICE B – CAPTURAS DE TELA DOS TESTES

Nesta seção de apêndices serão elencados os testes efetuados utilizando a aplicação teste que implementa as funções da biblioteca desse escrito. Na Figura 16 podem ser observados os valores na captura de 1 metro de distância.

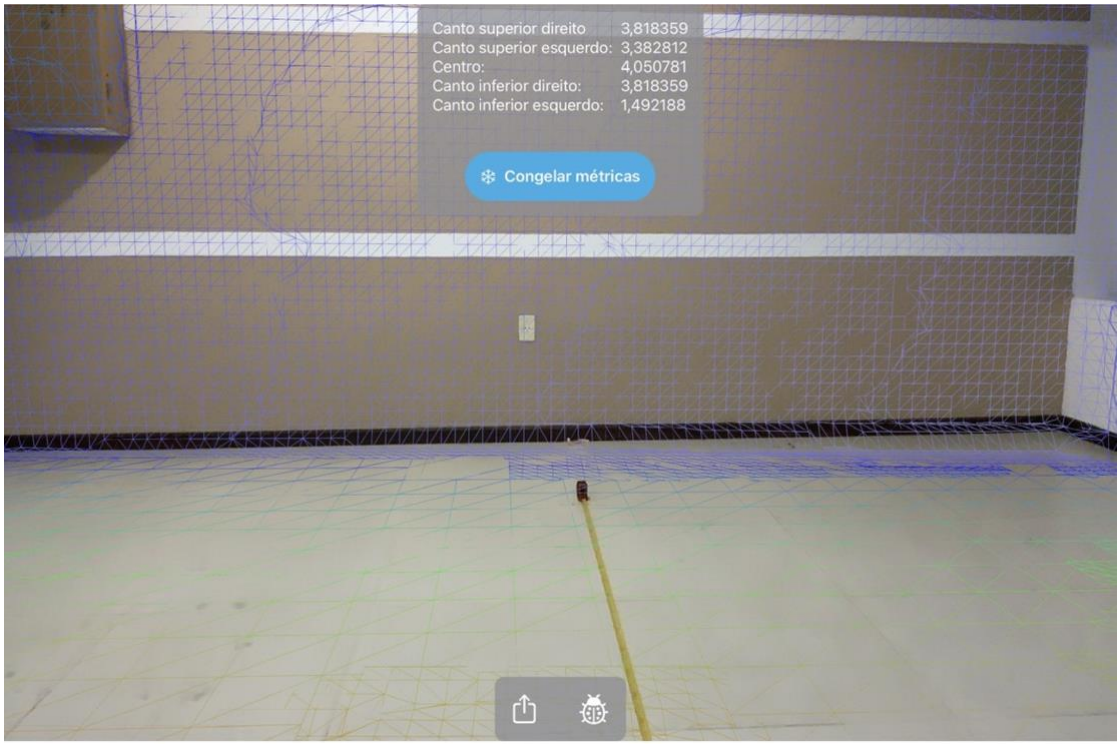
Figura 16 – Captura de 1 metro.



Fonte: elaborado pelo autor.

Na Figura 17 é possível observar o para a distância de 4 metros.

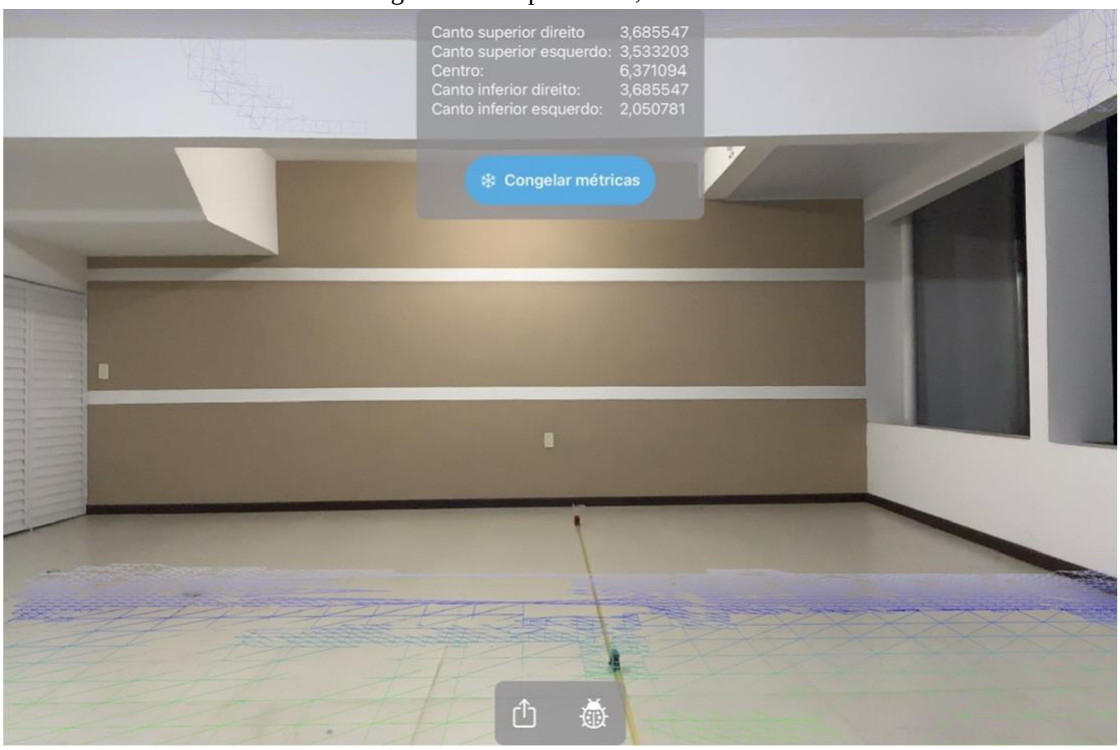
Figura 17 – Captura de 4 metros.



Fonte: elaborado pelo autor.

Por fim, na Figura 18, é possível observar a captura de 6,91 metros e vale considerar que as casas decimais todas variam constantemente não estabilizando em uma medição e que essa distância demonstrada foi congelada para fazer a captura de tela.

Figura 18 – Captura de 6,91 metros.



Fonte: elaborada pelo autor.