

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

Estudante: Gabriel Luís Fernando Vieira de Souza

Orientador: Dalton Solano dos Reis



Roteiro

Introdução

Objetivos

Fundamentação Teórica

Especificação

Implementação

Resultados

Conclusões

Sugestões

Apresentação Prática



Introdução

Se trata do processo de utilizar um sensor para mapear as distâncias de um ambiente e transformar elas em um modelo tridimensional.

LiDAR

Light Distance And Ranging

mede distância analisando a energia enviada e posteriormente retornada.

Digitalização 3D

O LiDAR acrescenta como um dos sensores disponíveis para ser utilizado no ARKit.

ARKit

Contextualização Histórica

Sensor de medição de distância.
Mapeamento Florestal de terreno.
Preservação histórica.



Objetivos

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.

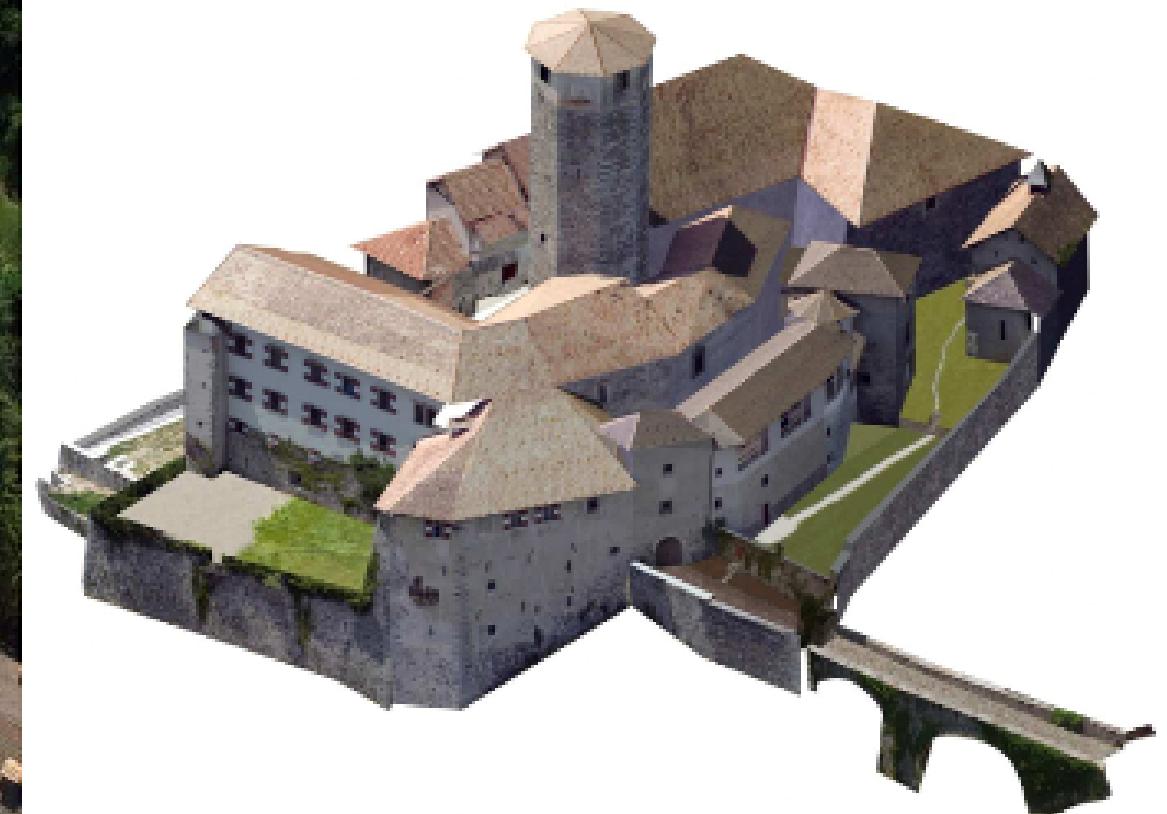
Objetivos Específicos

- 1 - Permitir o desenvolvimento de aplicações com digitalização 3D.
- 2 - Construir abstrações à tecnologia LiDAR para a digitalização 3D com as bibliotecas do ARKit.

Fundamentação Teórica (1/3)

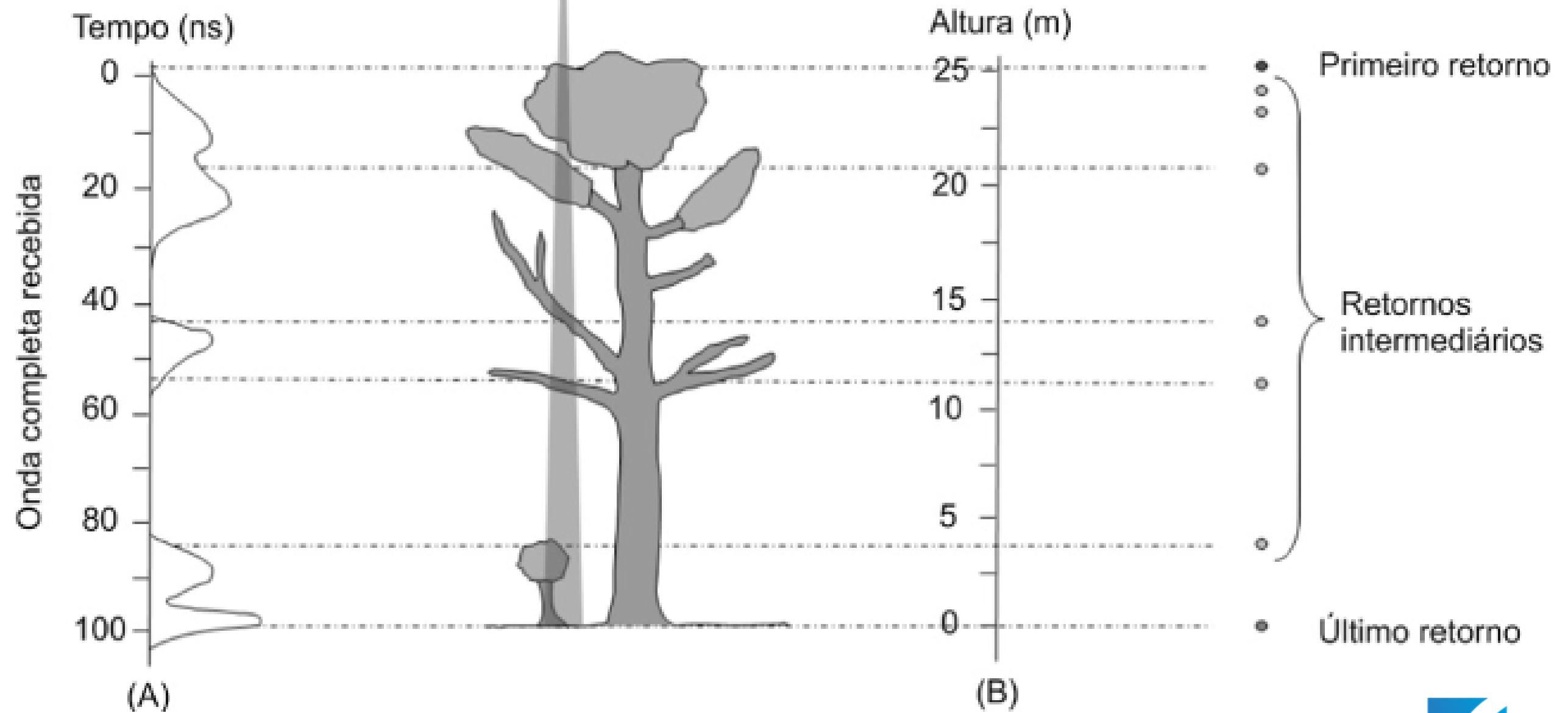
Digitalização 3D

- Modelagem 3D.
- Auxilio de sensores.
- LiDAR como modelador.



Fonte: Gonzo et al. (2007).

Fundamentação Teórica (2/3)



Fonte: Giongo et. al. (2010).

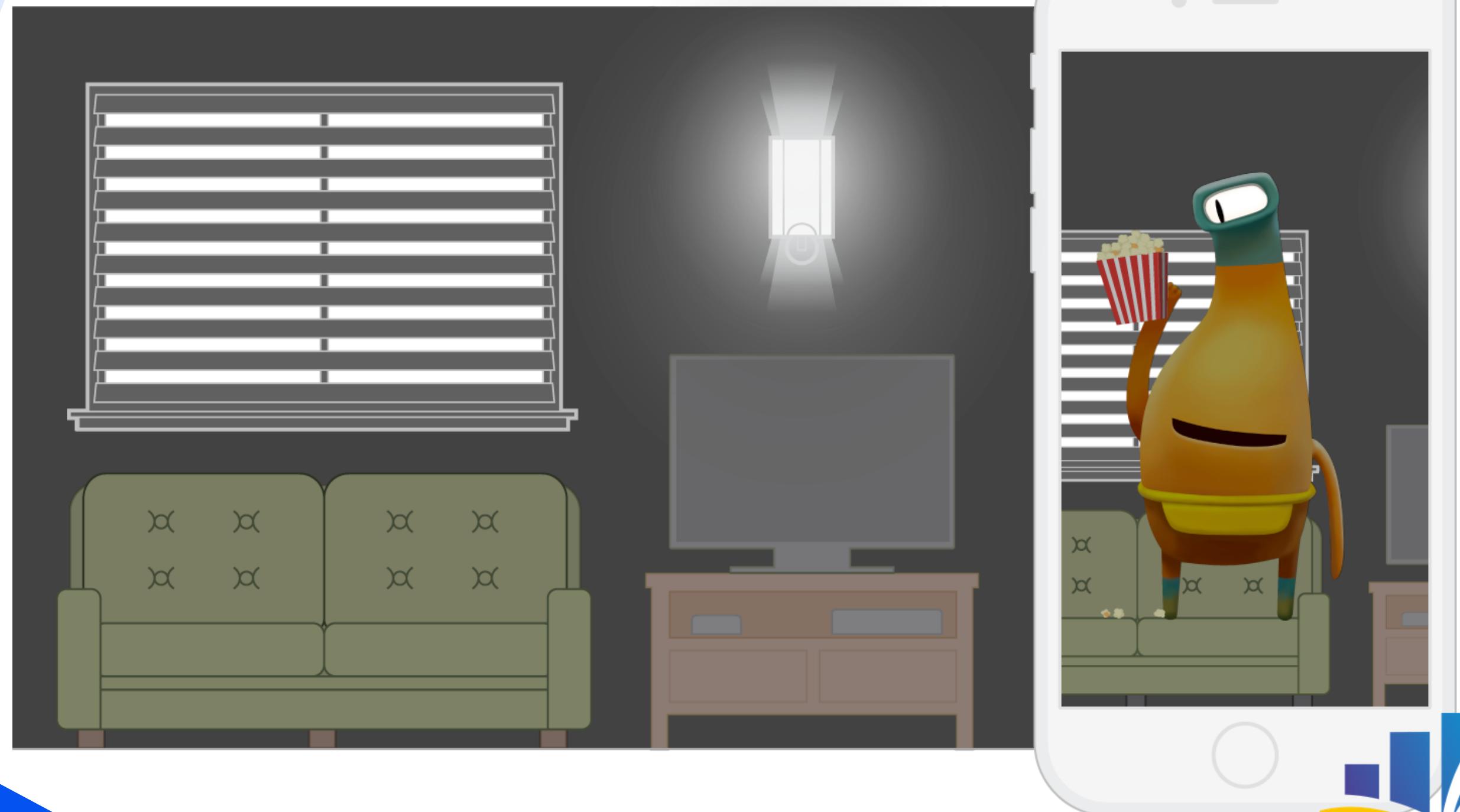
LiDAR

- Sensor móvel.
- Sensor fixo.
- Apple.

Fundamentação Teórica (2/3)

ARKit

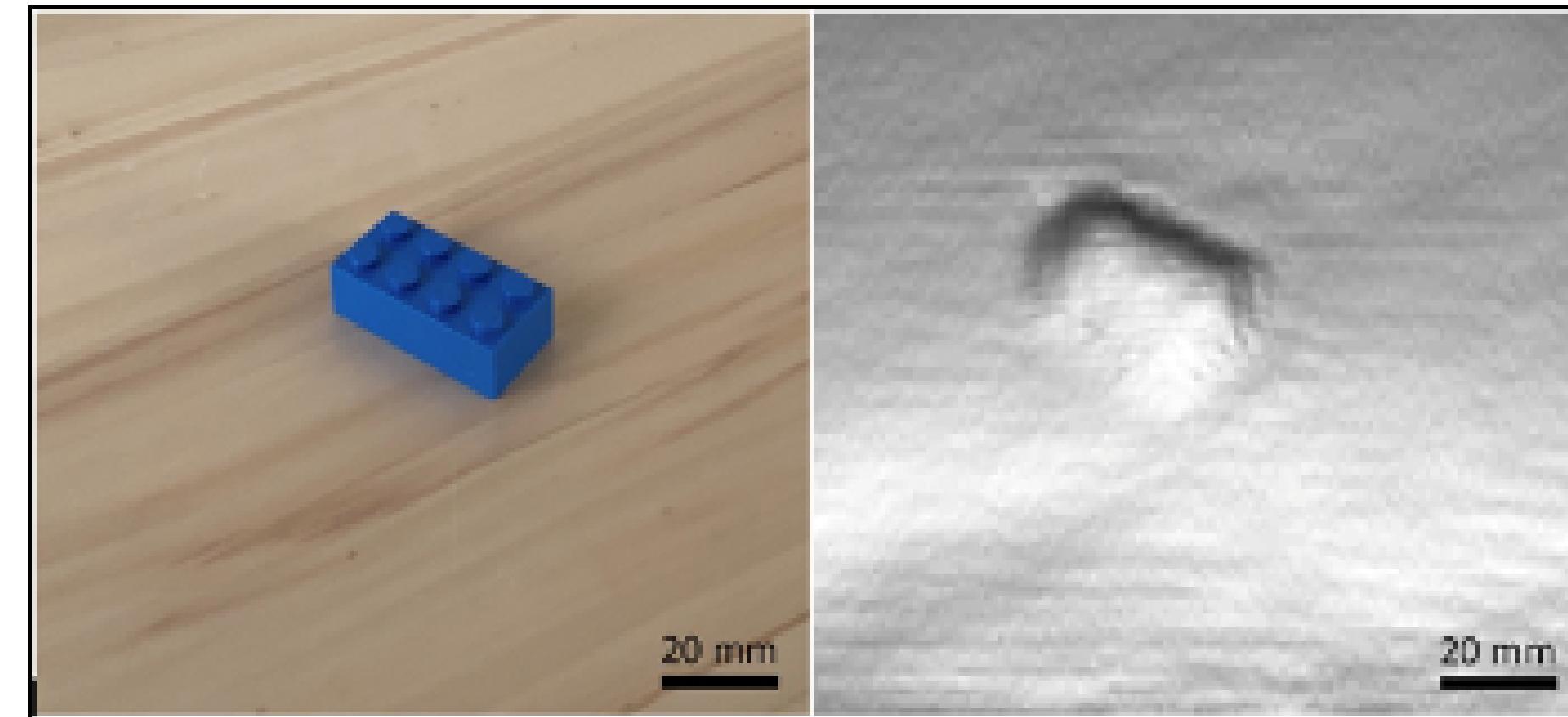
- Depth API.
- Scene Geometry.
- Instant AR.



Fonte: Apple (2022b).

Trabalhos Correlatos (1/3)

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



Fonte: Vogt, Rips e Emmelmann (2021).

Avalia as tecnologias LiDAR e TrueDepth disponíveis no iPad Pro em suas capacidades na digitalização tridimensional e compara com sensores industriais.

Trabalhos Correlatos (2/3)

Reconstrução 3D de Objetos com Kinect e Câmera Digital

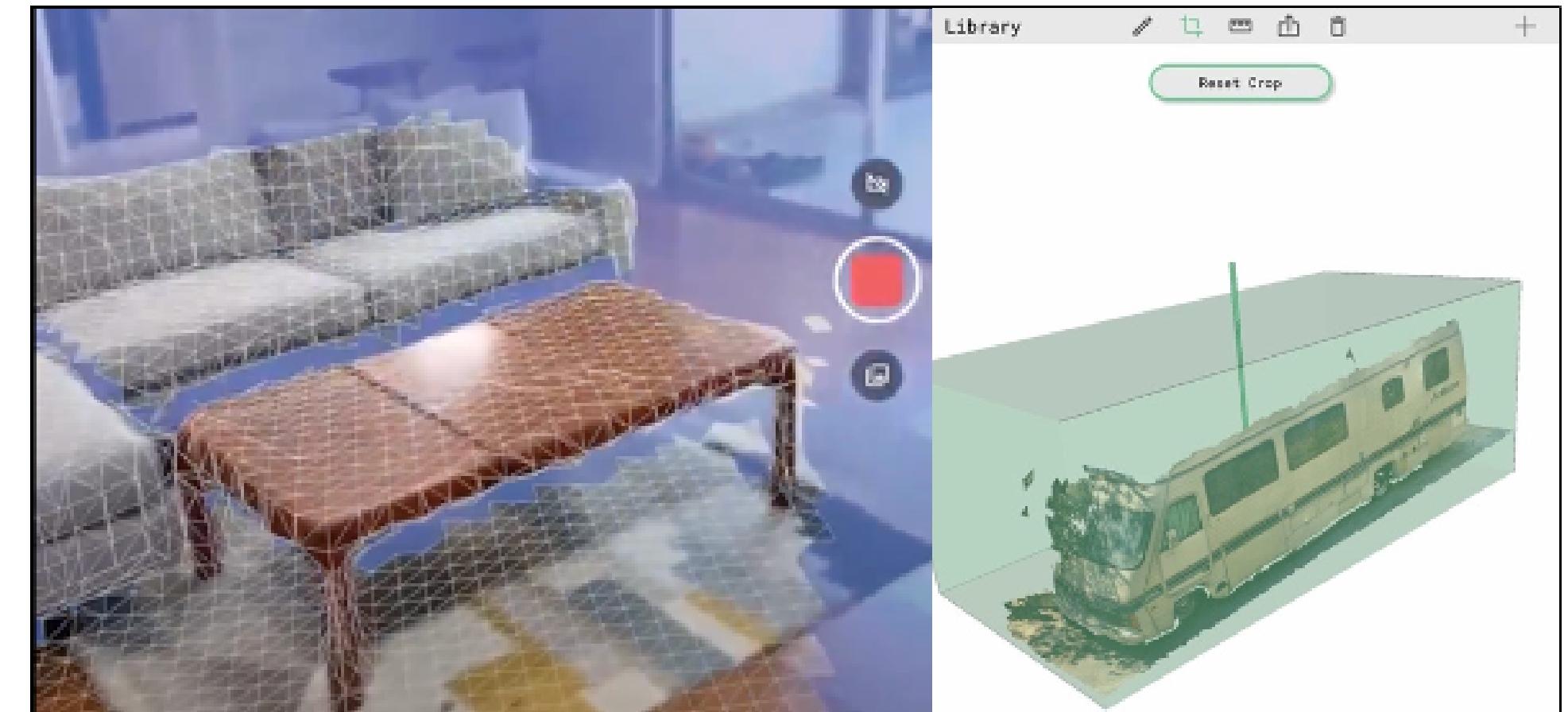


Fonte: Loura et al. (2018).

Realiza a digitalização 3D para preservação histórica de artigos de museu utilizando um Kinect e uma câmera digital.

Trabalhos Correlatos (3/3)

Polycam

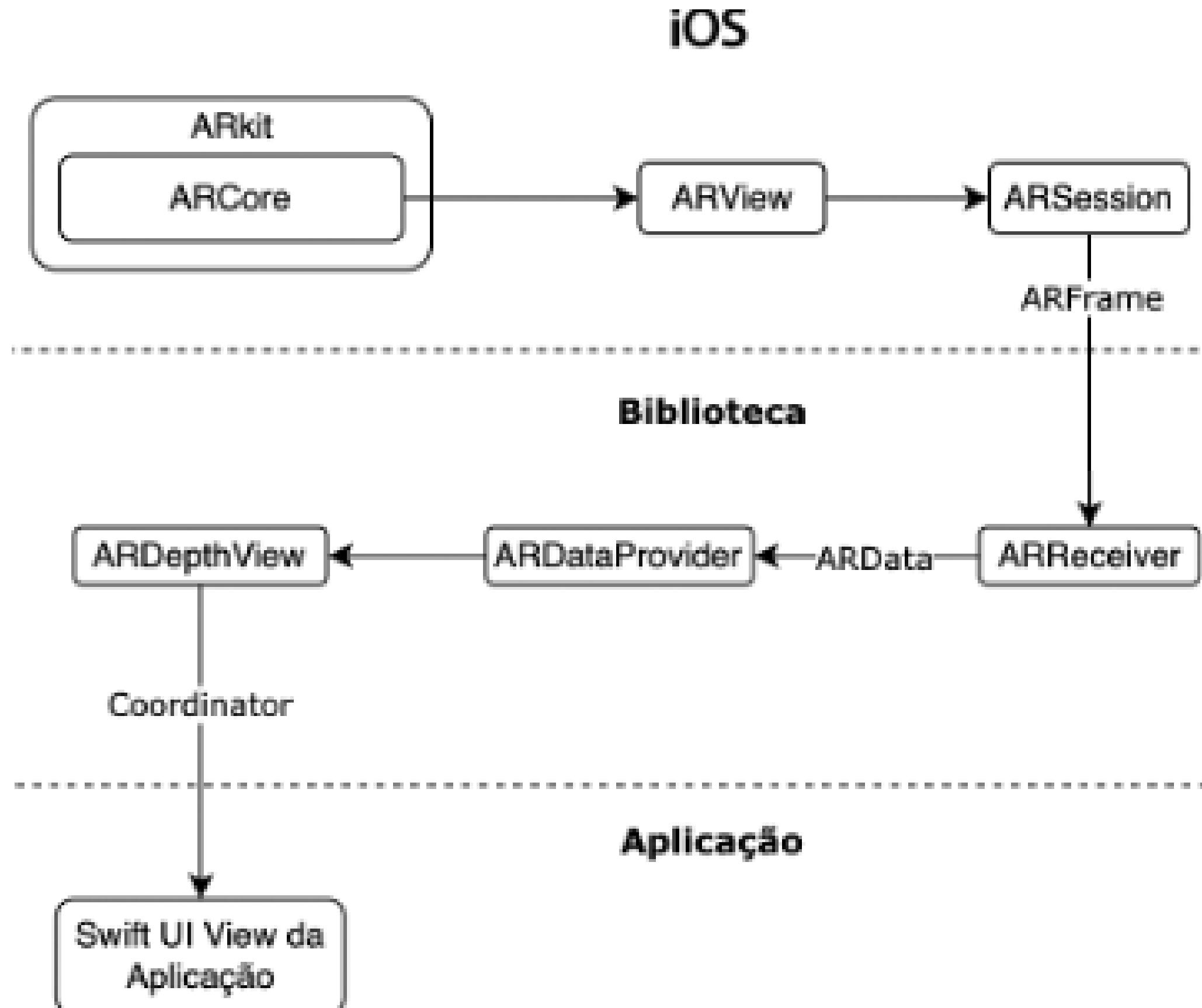


Fonte: Polycam (2021).

Gera modelos 3D a partir de escaneamento do ambiente utilizando o LiDAR e permite a exportação como um CAD.

Especificação (1/2)

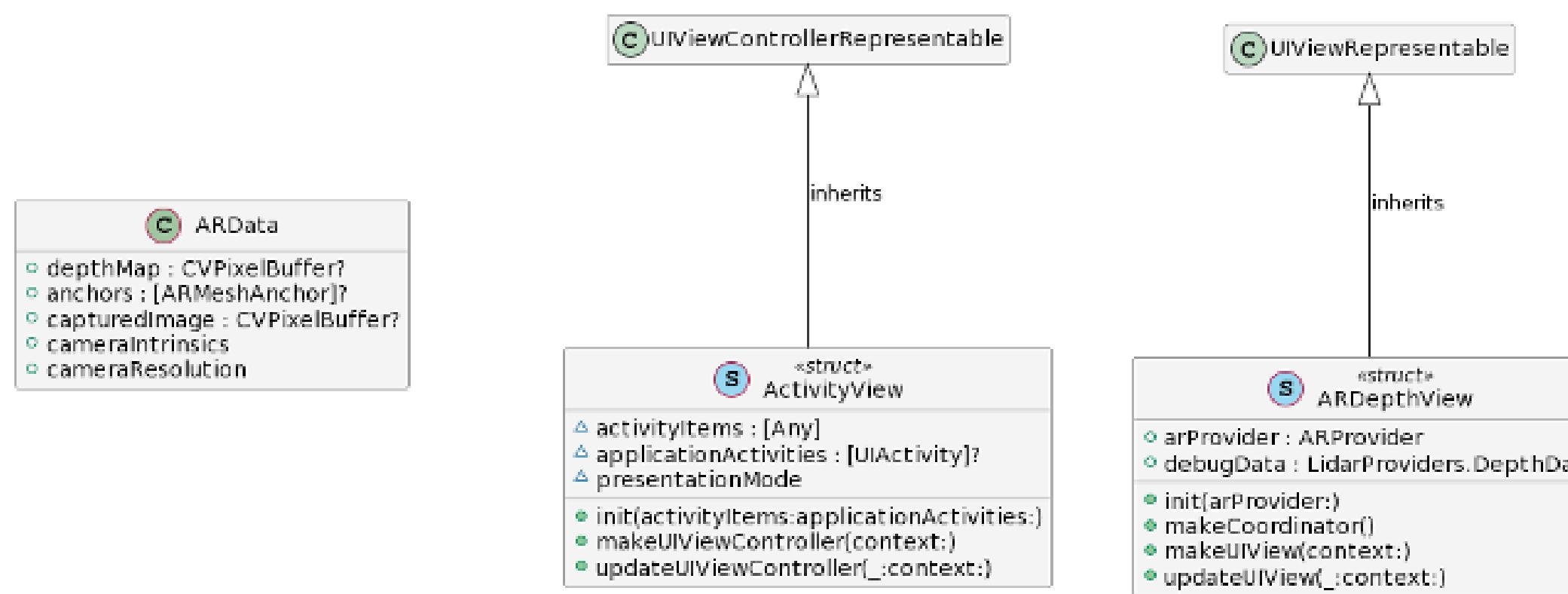
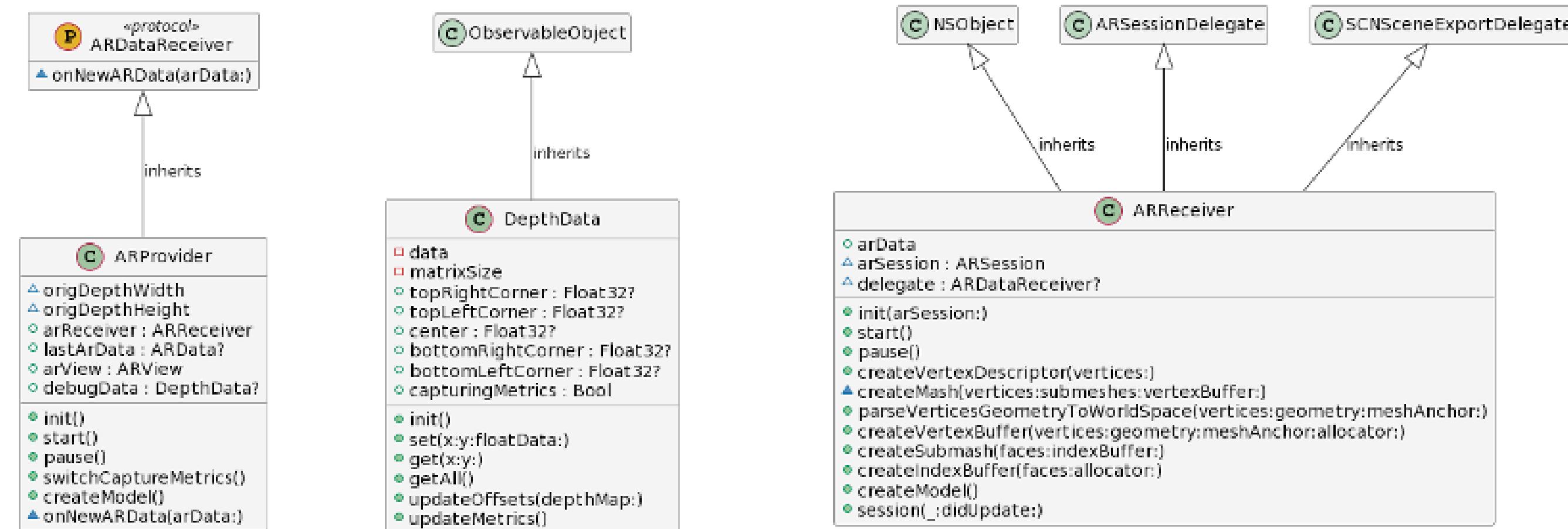
Diagrama de fluxo.



Fonte: Elaborado pelo autor.

Especificação (2/2)

Diagrama de classes.



Implementação (1/4)

```
22     public func makeUIView(context: UIViewRepresentableContext<ARDepthView>) -> ARView {  
23         let arView = arProvider.arView  
24  
25         // O coordinator do parâmetro context carrega uma instância de ARReceiver,  
26         // que é atribuído para a session do ARView, logo, cada atualização no  
27         // ARSession será repassada para o ARReceiver.  
28         arView.session.delegate = context.coordinator  
29  
30         arView.environment.sceneUnderstanding.options = []  
31         // Liga a oclusão da cena para a reconstrução da mesh.  
32         arView.environment.sceneUnderstanding.options.insert(.occlusion)  
33         // Turn on physics for the scene reconstruction's mesh.  
34         arView.environment.sceneUnderstanding.options.insert(.physics)  
35         // [OPCIONAL] demonstra o mapeamento da mesh na visualização  
36         arView.debugOptions.insert(.showSceneUnderstanding)  
37         // Desabilita as opções pois não são necessárias  
38         arView.renderOptions = [.disablePersonOcclusion, .disableDepthOfField, .disableMotionBlur]  
39         // Desabilita a configuração automática da sessão  
40         arView.automaticalyConfigureSession = false  
41  
42         return arView  
43     }
```

Fonte: Elaborado pelo autor.



Implementação (2/4)

```
122  public func createModel() -> SCNScene {  
123      // Nessa etapa é criado um alocador de memória, para guardar os  
124      // dados do objeto 3D até ser persistido  
125      let allocator = MTKMeshBufferAllocator(device: EnvironmentVariables.shared.metalDevice)  
126      // Com o alocador criado, um asset que irá montar o objeto 3D é criado  
127      let asset = MDLAsset(bufferAllocator: allocator)  
128      // aqui é feita uma verificação para garantir que  
129      // a instancia de ARData está preenchida com as ancoras do ambiente  
130      guard let anchors = self.arData.anchors else { fatalError("No anchors were found") }  
131  
132      // Então uma iteração é feita pelas ancoras encontradas  
133      for meshAnchor in anchors {  
134          // Recupera as geometrias da cena AR  
135          let geometry = meshAnchor.geometry  
136          let faces = geometry.faces  
137  
138          // Cria um buffer de índices que serão utilizadas como  
139          // a geometria do objeto 3D.  
140          let indexBuffer = createIndexBuffer(faces: faces, allocator: allocator)  
141  
142          // e adiciona em uma sub mesh  
143          let submesh = createSubmesh(faces: faces, indexBuffer: indexBuffer)  
144  
145          // Cria os vertices que conectam as faces e apartir deles  
146          // cria a mesh principal que será adicionada ao modelo 3D  
147          let vertices = geometry.vertices  
148          let vertexBuffer = createVertexBuffer(vertices: vertices, geometry: geometry, meshAnchor: meshAnchor, allocator: allocator)  
149          let mesh = createMash(vertices: vertices, submeshes: [submesh], vertexBuffer: vertexBuffer)  
150  
151          // Adiciona a mesh gerada ao modelo 3D  
152          asset.add(mesh)  
153      }  
154  
155      return SCNScene(mdlAsset: asset)  
156  }
```

Fonte: Elaborado pelo autor.

Implementação (3/4)

```
import SwiftUI
import LidarProviders

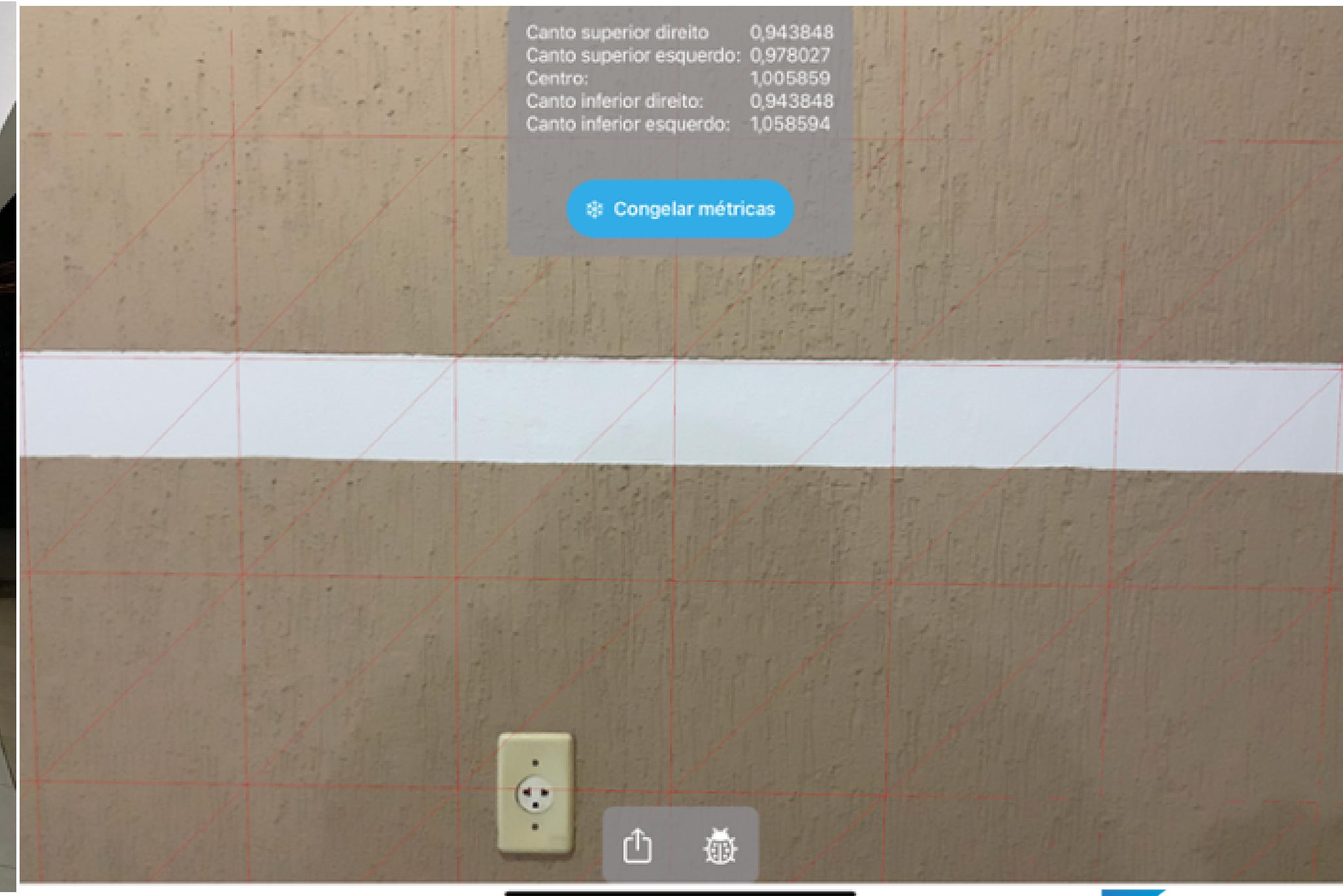
struct ScanView: View {
    let arProvider = LidarProviders.ARProvider()

    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.



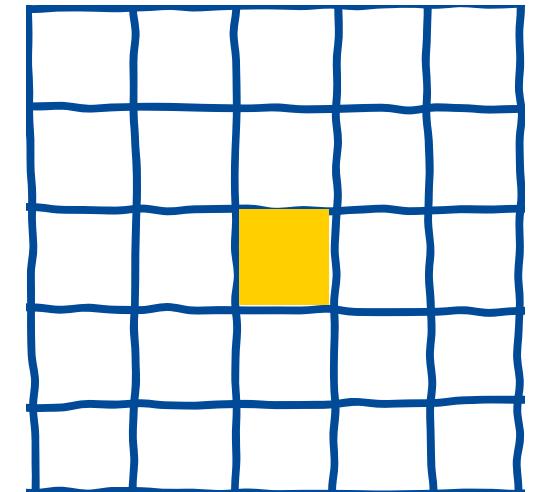
Implementação (4/4)



Fonte: Elaborado pelo autor.

Resultados

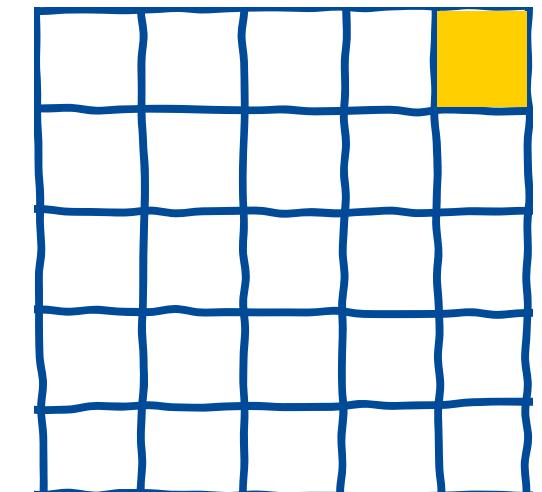
Medição da distância no ponto central detectado pelo sensor



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.

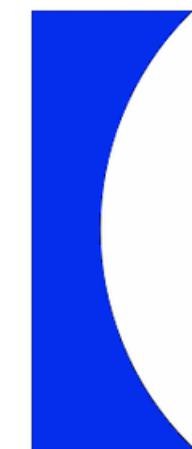
Medição da distância no ponto lateral superior direito detectado pelo sensor



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.

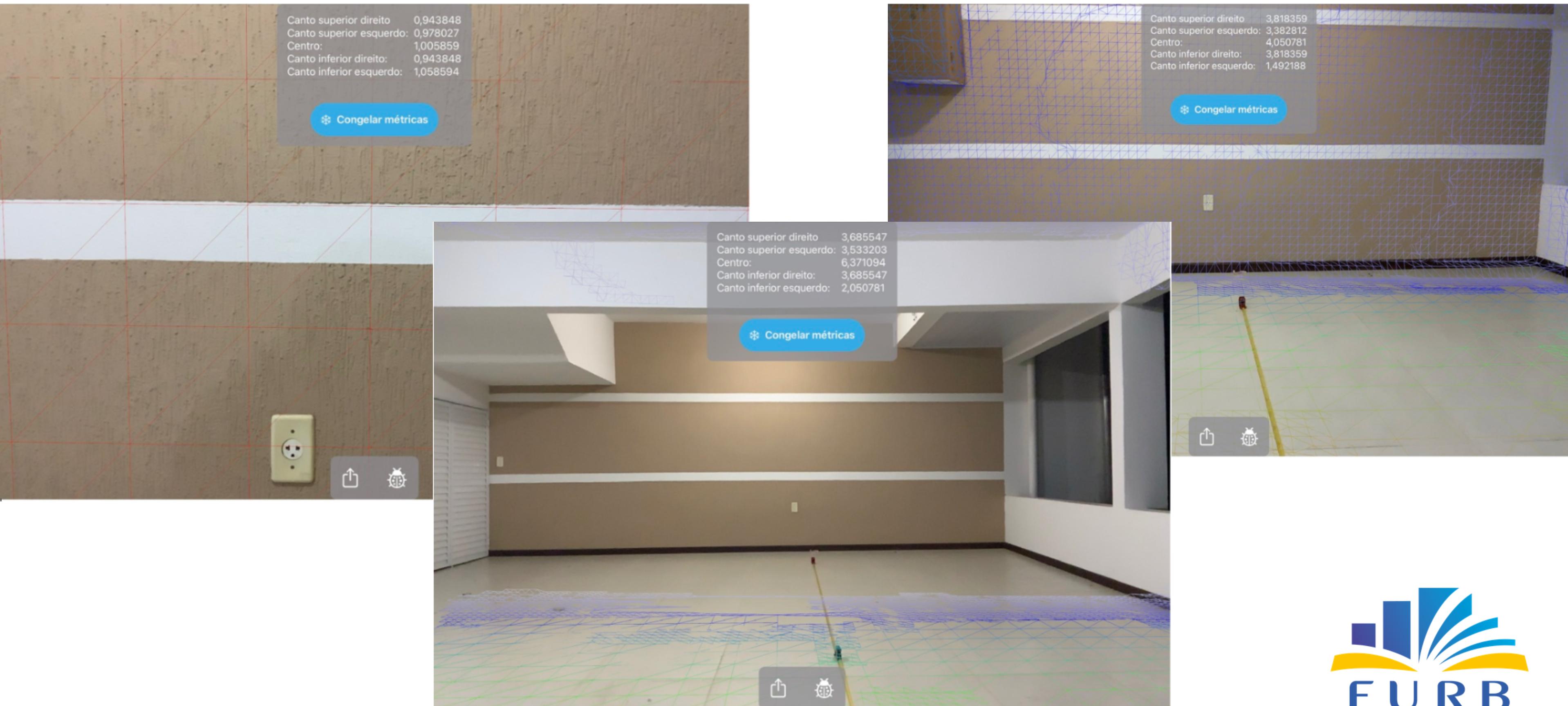
Medição da distância no ponto lateral superior direito detectado pelo sensor



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	-	-

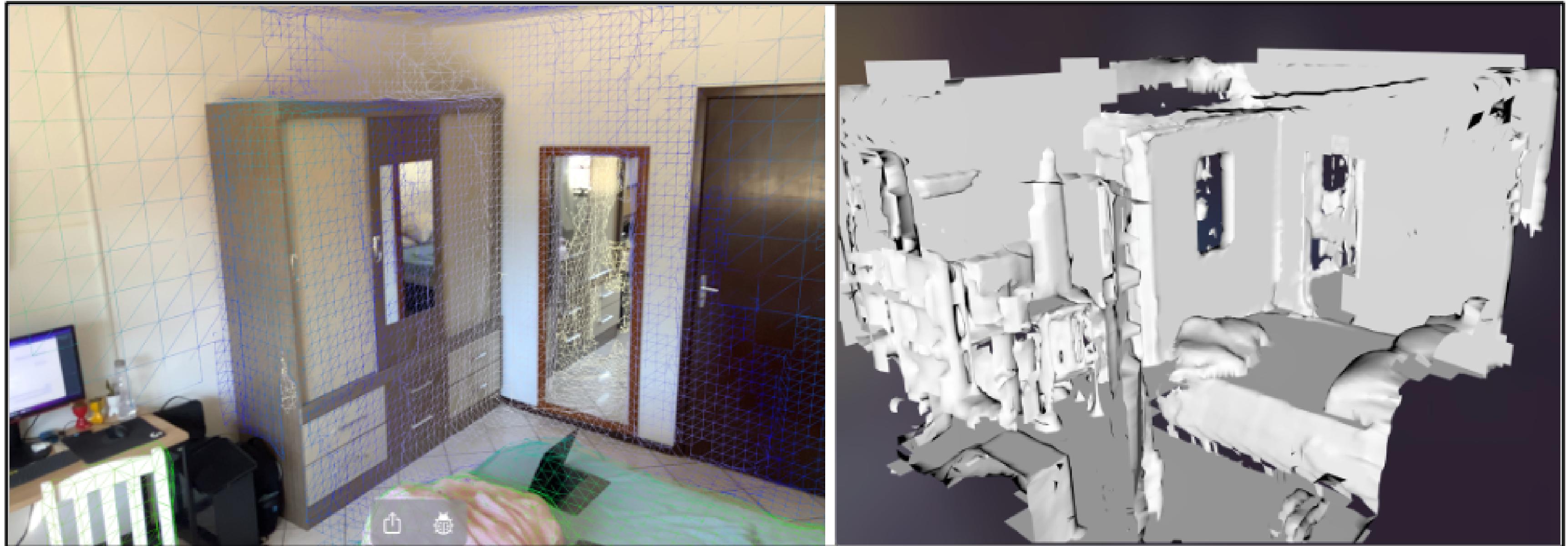
Fonte: Elaborado pelo autor.

Capturas de tela do aplicativo



Fonte: Elaborado pelo autor.

Escaneamento 3D de um quarto



Fonte: Elaborado pelo autor.

Conclusões

- Escaneamento 3D se mostrou possível mas com uma precisão baixa.
- Medição de distância se demonstrou precisa dentro da limitação de 5 metros prevista pela Apple.
- A biblioteca cumpre seu papel de disponibilizar APIs para digitalização 3D.

Sugestões

- Texturizar modelos 3D utilizando as informações de imagem da câmera.
- Criação de novos cenários de teste que envolvam movimento e diferentes tipos de materiais no escaneamento e medição.
- Utilizar a biblioteca como base para novos estudos envolvendo digitalização 3D.



Apresentação Prática

Obrigado!

Gabriel L. F. Vieira de Souza
gabrielluisfsouza@gmail.com

