INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




# Scan 3D de Interiores

**ANDRÉ FILIPE ROSÁRIO LIMA**
Junho de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

# Scan 3D de Interiores

**André Filipe Rosário Lima**

**Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas de Informação e Conhecimento**

**Orientador: Joaquim Peixoto**

**Coorientador: José Silva**

Porto, junho 2021

# Resumo

Este documento serve para apresentar o trabalho realizado no âmbito da disciplina TMDEI do Mestrado em Engenharia Informática (MEI) do Departamento de Engenharia Informática (DEI) do Instituto Superior de Engenharia do Porto (ISEP) do Politécnico do Porto (P. Porto).

Este trabalho foi realizado na empresa DevScope tem o intuito de analisar e avaliar um possível projeto que vise a implementação de um sistema de reconstrução/recriação e otimização de modelos 3D que represente os interiores de habitações em cenários virtuais.

Para a realização do projeto foi realizada pesquisa sobre metodologias de processamento de dados espaciais 3D e recriação de superfícies digitalizadas a partir de um scan. Esta informação suportará a realização de uma prova de conceito.

Para isso foi também avaliado o negócio em questão e como é que a implementação de uma solução de recriação dos espaços traria valor à instituição.

Após a análise foi realizada e avaliada uma prova de conceito que permita a recriação dos interiores das habitações. Estas recriações devem ainda ser otimizadas de forma que estas possam ser visualizadas num browser.

**Palavras-chave**: Augmented Reality, Virtualization, 3D Scan, 3D Space Reconstruction.

# Abstract

This document serves to present the work carried out under the subject TMDEI of the Master's in Informatics Engineering (MEI) of the Informatics Engineering Department (DEI) at the Instituto Superior de Engenharia do Porto (ISEP) of the Porto Polytechnic (P. Porto).

This work was carried out at DevScope with the aim of analyzing and evaluating the possibility of a project focused at implementing a system to rebuild/recreate and optimize 3D models that represent the real-world interiors of habitational buildings into a virtualized scenario.

To perform the described project, research was carried out on 3D spatial data processing methodologies and recreation of scanned surfaces obtained from a scan. All this gathered information will support the proof of concept.

For further supporting, it was also evaluated the business at hands and how such a solution to recreate interior spaces would bring value to the institution.

That said, a proof of concept will be carried out and evaluated which allows for recreation of homes interiors. This recreation should also be optimized for them to possibly be imported into a browser.

**Keywords**: Augmented Reality, Virtualization, 3D Scan, 3D Space Reconstruction.

# Agradecimentos

Agradeço em primeiro lugar à minha família e amigos que sempre me apoiaram e acreditaram no meu sucesso e pelo apoio que sempre me deram.

Também um agradecimento a todos os professores e colegas com quem me deparei ao longo do meu percurso académico, particularmente ao meu orientador Joaquim Filipe Santos, pelo apoio e suporte que me facultou durante todo este processo.

Agradeço às pessoas com quem me cruzei na empresa *DevScope* e aos seus colaboradores pelo acolhimento e apoio disponibilizado durante todo o projeto, com especial foco ao Luís Maia, David Mota e José António Silva.

# Índice

# Lista de Figuras

# Lista de Tabelas

# Acrónimos e Símbolos

## Lista de Acrónimos

| | |
|---|---|
| **AR** | Realidade Aumentada (*Augmented Reality*) |
| **CAD** | *Computer Aided Design* |
| **CNN** | Rede Neural Convolucional (*Convolutional Neural Network*) |
| **FAST Diagram** | Functional Analysis System Technique Diagram |
| **FPS** | *Frames por segundo* |
| **Imagem RGBD** | Imagem *Red Green Blue Depth* |
| **LiDAR** | Luz, Deteção e Alcance (Light Detection and Ranging) |
| **RANSAC** | *RANdom SAmple Consensus* |
| **SLAM** | Localização e mapeamento simultâneo **(***Simultaneous Location And Mapping*) |
| **VR** | Realidade virtual (*Virtual Reality*) |

# 1 Introdução

No presente capítulo introdutório, são apresentados o contexto, problema, objetivo, metodologia e estrutura do documento.

## 1.1 Contexto

A evolução e disponibilidade de tecnologias de realidade virtual e aumentada (VR/AR) está a tornar-se cada vez mais popular, numa ótica em que os utilizadores se conectam de uma forma mais convicta e de forma a obter uma perceção melhorada da mensagem pretendida pela aplicação [1].

Existindo atualmente implementada uma plataforma que renderiza imagens 360º, estas imagens são apenas uma aproximação do real, sendo que são passados pontos vetoriais 3D para uma visualização 2D dos interiores que são pretendidos dar a conhecer aos utilizadores.

Deste modo, existe uma vontade de evoluir, criando uma perspetiva mais precisa da realidade tirando partido das tecnologias de AR existentes. Um exemplo de possível solução é a utilização de captura a partir de smartphones, cujas capacidades computacionais foram aumentando nos últimos anos [2].

Outro aspeto importante é a acessibilidade pelo utilizador a dispositivos de scan (como o LiDAR no caso do iPhone12 Pro) que outrora estavam restritos a profissionais da área ou necessitavam de um forte investimento financeiro para a sua aquisição.

Assim sendo, o problema deparado é a vontade de analisar a possibilidade em melhorar a perceção obtida pelos clientes no contacto com os interiores de habitações. Este problema é analisado mais pormenorizadamente de seguida.

## 1.2 Problema

Na solução atual para a recolha e apresentação dos interiores de habitações a clientes, são recolhidas fotografias dos interiores, estas fotografias são processadas e apresentadas num website numa ótica de vista 360º.

Para a realização da vista 360º, é necessário haver pelo menos 4 fotografias ou uma fotografia panorâmica, que são colocadas nas quatro paredes da habitação virtual. Esta metodologia apesar de uma aproximação à realidade, continua a ser uma abordagem 2D para a apresentação do interior das casas, uma vez que falta a noção de profundidade no cenário apresentado aos clientes.

## 1.3 Objetivo

O objetivo pretendido nesta dissertação é o desenvolvimento de uma prova de conceito que permita realizar a recriação 3D de interiores de habitações. Uma vez que a solução existente utiliza fotografias panorâmicas para recriar o interior das habitações, esta dissertação tem o objetivo de ir mais além e recriar as habitações através do scan.

Teoricamente através do scan é possível obter uma definição mais elevada diretamente a partir da informação obtida pelos sensores. Assim sendo, a melhoria da solução existente e evolução da mesma de uma fotografia 360º para um cenário virtual é o que é desejado testar com a prova de conceito.

Deste modo, a prova de conceito deverá tentar responder às seguintes hipóteses:

- É possível recriar o interior das habitações na sua totalidade e com qualidade?
- Qual a melhor estratégia para recriar espaços interiores?

## 1.4 Método

Para concretizar os objetivos descritos anteriormente, é necessário pesquisar e estudar o estado da arte relacionado com o scan 3D e renderização de cenários virtuais, principalmente na reconstrução de *point clouds* em modelos 3D.

Após realizado o estudo, são analisadas as opções e escolhidas aquelas que são mais pertinentes e que melhor se enquadram no contexto, bem como as que permitem obter maior valor para a organização.

Tendo escolhido as tecnologias e realizado o design da aplicação, procede-se à implementação da prova de conceito da mesma, tendo em conta boas metodologias e padrões de desenvolvimento.

## 1.5 Estrutura do documento

O presente capítulo, Introdução, tem a finalidade de facultar uma compreensão inicial do documento e a sua respetiva estrutura.

O capítulo 2, Estado da arte, tem no seu conteúdo os conceitos e o levantamento do estado da arte referente a tecnologia relevante.

O capítulo 3, Análise de negócio, tem como objetivo apresentar o tema do projeto de uma forma mais pormenorizada, os processos e intervenientes envolvidos e as restrições encontradas na implementação da solução.

O capítulo 4, Análise de valor, é possível encontrar uma análise descritiva do valor do projeto considerando os respetivos processos.

O capítulo 5, Desenho da Solução, foca-se na escolha e das possíveis abordagens alternativas e do desenho ao nível arquitetural do problema.

O capítulo 6, Construção da solução, são apresentados os métodos e a forma como foi construída a solução para o problema, tendo em conta os detalhes da implementação e prova de conceito.

O capítulo 7, Experimentação e Avaliação, é realizada uma comparação entre as soluções existentes (segundo fontes de informação definidas) e a solução obtida.

O capítulo 8, Conclusões, é realizado um levantamento dos objetivos alcançados, as limitações encontradas durante a realização do projeto e as possibilidades para trabalho futuro.

No final do documento, é possível encontrar as Referências utilizadas no documento aqui presente.

# 2 Estado da arte

Neste capítulo são apresentados contextos chave para a compreensão do problema e resolução do mesmo, tendo ainda algumas abordagens já efetuadas a questões da mesma área.

## 2.1 Representação da informação

Existem várias formas de obter e armazenar a informação obtida pelos sensores. Numa primeira instância, a informação acerca do mundo é capturada e armazenada numa *point cloud*, através de imagens RGB ou imagens de profundidade. Os dados podem ficar sobre esta forma ou serem processados e armazenados numa forma mais organizada, sendo que é o seu pré processamento é, por norma, recomendado.

### 2.1.1 Point cloud

"A *point cloud* é normalmente formada por um grande conjunto de dados, onde as várias coordenadas são redundantes"[1]. Antes de utilizar os dados, é recomendado que seja feita algum tipo de pré-processamento dos mesmos. [3]

A *point cloud* é a data extraída com um scanner e tem como propósito facilitar a compreensão das coordenadas do mundo. Estas nuvens são um conjunto de dados referentes a pontos num sistema de coordenadas de três dimensões.

O principal objetivo deste tipo de estrutura de dados é representar a superfície dos objetos e não o de conter dados sobre a sua cor, textura ou material. Estas últimas caraterísticas podem

[1] Texto adaptado do inglês: "Point clouds are usually formed by a large amount of data, where many coordinates are redundant." [3]

ser obtidas através de software CAD que traduz as *point clouds* (com auxílio a imagens) a figuras geométricas com a informação adicional. [4]

Existem ainda processos para modelar este tipo de dados: [4]

- Processamento paramétrico baseado em caraterísticas e histórico sendo que caso uma caraterística seja referenciada no futuro, apagar ou modifica a mais antiga interfere com as referências mais recentes.

- Modelação direta não referencia o histórico das alterações e permite a edição das caraterísticas em tempo real. É mais rápida que o processamento paramétrico.

- Modelação baseada em malha que assenta na utilização de polígonos para modelar os dados (sendo que os triângulos são os polígonos mais utilizados), tem a utilidade de recriar modelos 3D e superfícies. Através deste método é ainda possível converter com maior facilidade as *point clouds* em malha poligonal, sendo que o seu nível de definição e qualidade é proporcional ao poder computacional utilizado.

### 2.1.2 Voxel

No espaço 3D, o volume pode ser dividido em linhas e colunas, uniformemente espaçadas entre si, dividindo assim o espaço em cubos. O termo *voxel* é o nome dado a esses cubos, portanto um *voxel* está para um espaço 3D como o *pixel* está para o 2D. Cada *voxel* é definido com coordenadas 3D e pode ter uma cor associada a si. [5]

#### 2.1.2.1 Voxels e classificação de regiões 3D

Os *voxels* tem utilidade para catalogar regiões do mundo 3D uma vez que cada elemento no mundo virtual é um *voxel*. Cada *voxel* tem 26 vizinhos, o que permite definir regiões através do agrupamento de *voxels* com o mesmo valor (número de região ou de cor) presentes em cada um. Este tipo de agrupamentos permite reconhecer os limites de superfícies e objetos 3D devido ao contraste entre cores e texturas nos *voxels* vizinhos (visto que *voxels* da mesma superfície/objeto tendem a ter cores e/ou texturas similares). [6]

#### 2.1.2.2 Octree

As octree são estruturas hierárquicas de dados que são utilizadas em espaços 3D para os particionar em *voxels* de tamanho variado. As octrees são uma ferramenta recursiva para criar subdivisões de um universo finito de cubos, facilitando assim a reconstrução de modelos dentro desse espaço dividido. As octrees são uma representação hierárquica em potências de base 2 ($2^n \ \wedge n \geq 1$), permitindo assim sucessivas partições do universo em voxels de menor dimensão. Um exemplo de uma possível partição referente a um cubo é encontrado na Figura 2-1. [7]

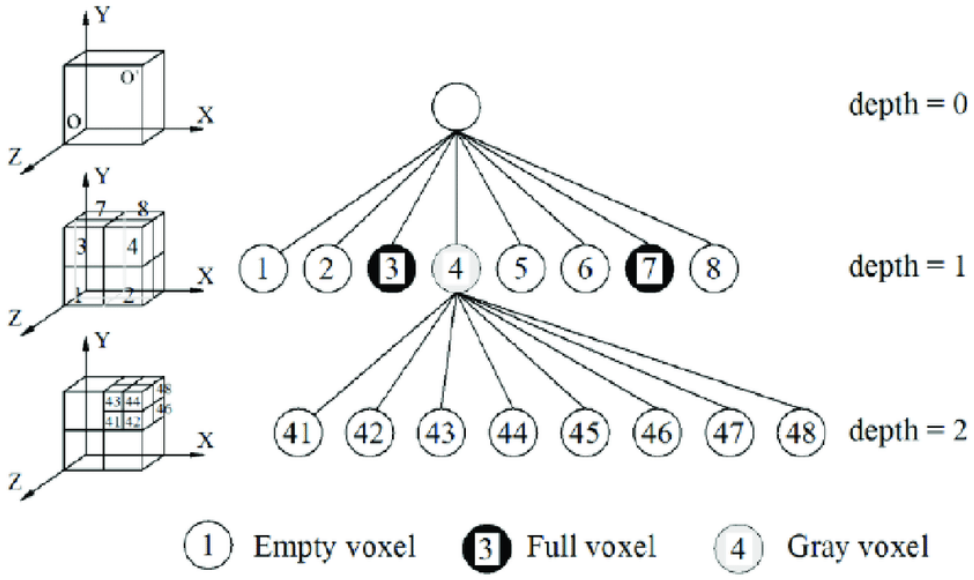


Figura 2-1 – Criação de octree a partir de partição de um cubo [8]

As principais caraterísticas do uso de octrees são: [7]

- É uma estrutura hierárquica adequada para representar dados 3D;
- Este estilo de organização de dados permite realizar operações de *drill-down* e *roll-up* sobre os dados, algo útil caso seja necessário explorar algum tipo de objeto;
- A forma como a octree é construída tem semelhanças com os processos de agrupamento na mineração dos dados.

### 2.1.3 Imagem RGB(D)

Imagem RBG é uma imagem com cor (daí R-*red*, G-*green*, B-*blue*). Este é o tipo de imagens mais convencionai, no entanto existe ainda a imagem de profundidade que permite identificar as distâncias de um certo ponto da imagem a um outro objeto da mesma.

Deste modo é possível obter ou formar imagens RGBD quando uma imagem de profundidade é associada a uma mesma imagem RGB.


Figura 2-2 – Exemplo de imagem RGB e profundidade. Na coluna (A) encontram-se as imagens originais, na (B topo) imagens percebidas (através de catalogação semântica). (B baixo) imagens de profundidade capturadas através de um sensor de profundidade. [9]

Através de imagens RGBD é possível obter informação acerca da cena real e traduzi-la para um mundo virtual. Esta abordagem documentada por Zou, Guo, Li & Hoiem [10] que utilizam a rotulagem 3D para analisar e traduzir a imagem para informação computacional necessária para renderizar a cena. Este tipo de abordagem requer um conjunto de dados para treino do algoritmo, de forma que este possa reconhecer e diferenciar os distintos objetos e distância destes para a câmara e entre os restantes objetos da cena.

As imagens são então processadas por um algoritmo de extração de caraterísticas *rbg&depth* para obtenção dos dados percecionados. Após a obtenção destes dados iniciais, esta informação é transmitida a um algoritmo de classificação que cataloga cada segmento da imagem que. Após a catalogação, os segmentos são associados a imagens pré-definidas e armazenadas para serem associadas a objetos com catalogação tipo. No final, os objetos reais são representados no mundo virtual no mesmo local, mas com objetos tipo armazenados. Este processo é descrito na Figura 2-3. [10]

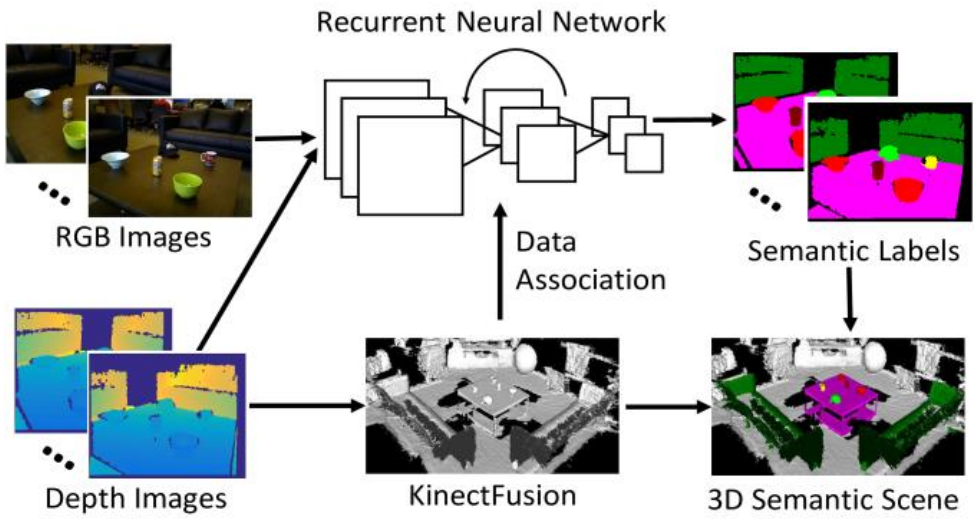


Figura 2-3 – Exemplo algoritmo de rotulagem 3D a partir de imagem [11].

### 2.1.4 Modelação em malha ou poligonal

A modelação em malha ou poligonal consiste em agrupar os pontos 3D do mundo em conjuntos de vértices e faces para que seja possível criar representações de superfícies e modelos 3D. As figuras poligonais podem ser construídas a partir de múltiplos objetos (segmentos) ou de uma simples malha poligonal. [12]

Para modelar objetos complexos com vários objetos e/ou formas que o compõem, estes são dispostos de forma hierárquica, através de articulações e segmentos rígidos. Isto significa que caso uma transformação for realizada numa parte do objeto, a restante parte hierarquicamente abaixo da que sofreu alteração irá sofrer essa mesma alteração também. [12]

As representações em polígonos ou malhas poligonais têm a sua melhor utilização quando a velocidade de renderização é crucial ao problema (em videojogos ou flexibilidade topológica da solução). Um problema deparado ao utilizar esta abordagem é a alta necessidade de poder de computação e um elevado conjunto de dados, para que a área a representar obtenha uma precisão e definição melhorada. [12, 13]

Quando é necessário manipular um elevado conjunto de modelos poligonais, existem duas abordagens principais: [13]

- Utilizar a *Mesh Simplification* para reduzir o número de polígonos que representam a superfície;

- Utilizar o *Geometry Culling* que reduz o número de pixels no *framebuffer*. Este pode ser feito de três formas:

- *Backface Culling* que remove a face traseira dos polígonos, regida através da normal associada à face;
- *Frustum Culling* que remove os objetos que não se encontrem parcialmente visíveis no *frustum* da vista;
- *Occlusion culling*, processo que elimina objetos que não são visíveis devido ao bloqueio da vista da câmara por outros objetos.

## 2.2 Localização do dispositivo no espaço

Para existir a possibilidade de recriar o espaço através dos dados obtidos, uma informação importante é a localização do sensor (que capturou os dados) relativamente aos objetos que se pretende digitalizar. O definir da localização da câmara no espaço é também importante para garantir uma maior imersividade no mundo virtual ao utilizador. Para isso, de seguida são apresentados a Odometria e o SLAM como resposta ao problema de localização.

### 2.2.1 Odometria

A Odometria é utilizada para rastrear a localização atual no espaço real (x, y, z). As coordenadas do utilizador são estabelecidas através da integração de um sensor que capta movimentos do dispositivo, calculando assim a diferença de local ao longo do tempo. Ao relacionar esta informação com o ângulo em que o dispositivo se encontra, é possível criar e estimar os seis graus de liberdade de um sistema.[2] [14]

#### 2.2.1.1 Mapeamento por pontos chave de um *frame*

Uma das metodologias para realizar a odometria é o mapeamento por pontos chave de um *frame*. Esta metodologia consiste em escolher pontos-chave do mundo, num certo *frame*, e comparar a localização relativa entre o dispositivo e esses mesmos pontos ao longo dos *frames*. Isto é, comparar a localização relativa aos pontos entre o *frame* atual e os anteriores, de forma a mapear o dispositivo no mundo (Figura 2-4).

[2] O grau de liberdade de um sistema refere-se às formas básicas de movimento que este pode realizar. Os seis graus são referentes aos movimentos de rotação (rodar esquerda/direita; inclinar cima/baixo; virar esquerda/direita) e de translação (mover frente/trás; lateralmente; verticalmente) sobre os eixos (x, y, z). [63]

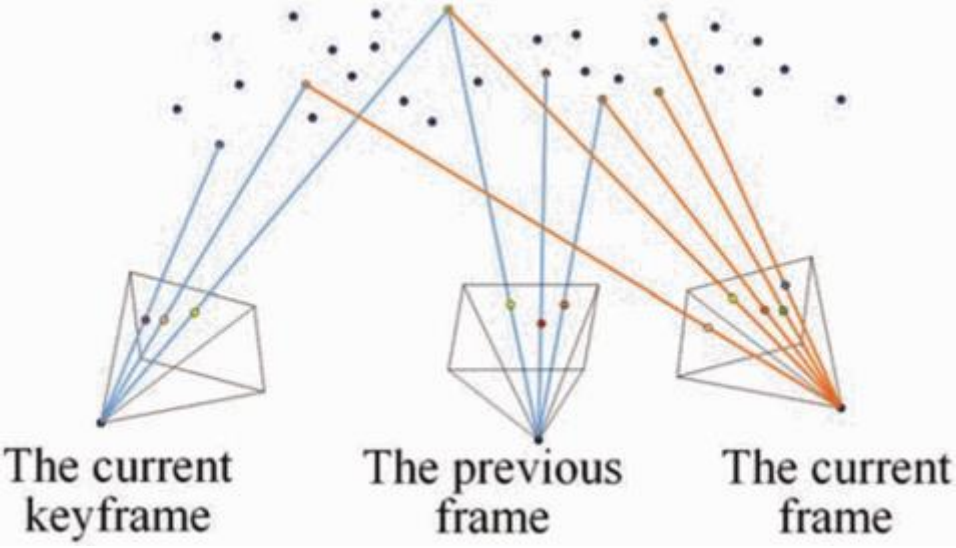


Figura 2-4 – Estimativa da localização do dispositivo através de pontos-chave [15]

Para realizar este método é necessário: [15]

- Obter informação relativa entre o dispositivo e os pontos-chave e guardar essa informação ao longo dos *frames*;
- Relacionar a localização relativa aos pontos chave na posição início e na atual, bem como a orientação e trajetória do dispositivo;
- Calcular a distância entre o dispositivo e os pontos-chave referenciados.

#### 2.2.1.2 Extração e correspondência de caraterísticas ORB

ORB (*Oriented FAST and Rotated BRIEF*) é a fusão entre o detetor de pontos-chave FAST[3] e o descritor BRIEF[4] com várias modificações de melhoria de performance. Numa primeira fase, é utilizado o FAST para encontrar os pontos-chave. Um problema com o FAST é que não calcula a orientação, sendo esta obtida através do descritor BRIEF dos pontos. [16]

### 2.2.2 SLAM Visual

SLAM é a terminologia dada ao processo de determinar a posição e orientação dispositivo em relação ao seu ambiente circundante. SLAM é também utilizado para calcular as trajetórias percorridas pelo sensor e criar um mapa a partir desses movimentos. [17, 18]

[3] *Features from Accelerated Segment Test* é uma metodologia para detetar cantos, podendo ser utilizada como identificador e extrator de pontos-chave. Pode ainda ser utilizado para rastrear e mapear objetos na visão computacional. Uma vantagem do FAST é a sua eficiência computacional. [64]

[4] *Binary Robust Independent Elementary Features* é um método para determinar descritores nos pontos caraterísticos. Estes pontos são principalmente utilizados para localizar o dispositivo no mundo, através do cálculo de distâncias euclidianas entre os pontos e o dispositivo. [65]

O SLAM Visual (VSLAM) é um tipo específico de SLAM que utiliza a componente visual 3D para realizar as tarefas de calcular a trajetória e tomar conhecimento das suas redondezas. [17]

Assim sendo, o SLAM resolve problemas como:

- **Sistema de coordenadas** – Como foi referido, um dos objetivos do SLAM é o cálculo da trajetória. Para isso, é necessário compreender os movimentos efetuados e as mudanças nas coordenadas que foram realizadas ao longo do tempo. Esta compreensão é obtida através da corelação entre os sensores e as mudanças de vistas do dispositivo. [18]

- **Localização –** Relativamente à localização do dispositivo, o SLAM utiliza pontos com coordenadas projetadas para definir o seu posicionamento e orientação no espaço. Para obter tal informação, o dispositivo mantém a informação de um conjunto de pontos ao longo de sucessivos *frame*s da imagem, de forma a triangular a posição 3D do dispositivo e dos objetos que o rodeiam. Esta informação é necessária para obter um posicionamento e orientação do dispositivo em relação à cena. [17, 18]

- **Mapeamento –** Um dos benefícios da utilização do SLAM (principalmente o VSLAM) é o de poder ser feita com alguma precisão a área por onde o dispositivo passou. São necessárias a localização e a pose do dispositivo. Após obter tais dados, é possível obter um mapa do ambiente em torno do caminho efetuado pelo dispositivo, através de um mapeamento relacional entre o caminho efetuada e as imagens capturadas pela câmara.

## 2.3 Abordagens para modelação 3D

Existem várias formas e abordagens para manipular e modelar os dados obtidos pelos sensores/câmaras, de forma a criar mundos virtuais a partir dos mesmos. De seguida são apresentadas algumas destas formas.

### 2.3.1 Segmentação por plano

A segmentação por plano é um processo de obtenção de superfícies a partir de uma *point cloud* 3D. Tal como o nome indica, os pontos da nuvem são agrupados por segmentos ou partes dessa mesma nuvem. Estas segmentações formam superfícies que correspondem às superfícies dos objetos reais. [19]

As superfícies planas obtidas não são uma definição geométrica de um plano, uma vez que um plano é geometricamente definido por um ponto e um vetor normal, enquanto as superfícies são um conjunto de pontos no mundo. As superfícies planas têm as vantagens de:

- Sendo um conjunto de pontos, estes são normalmente delimitados, o que dá uma melhor representação de um modelo real;

- A inexistência de pontos numa certa região de uma superfície indica que poderá existir um buraco naquela superfície;

- As superfícies podem ser computadas para um plano geométrico através de métodos *Best fit*;

- A margem dos planos pode ser obtida por um ajuntamento de pontos ao longo de um sítio.

Apesar das suas vantagens, a segmentação por plano requer uma elevada utilização de memória e uma sobrecarga computacional. Um conjunto de pontos planares pode ser denso e conter informação redundante acerca da superfície. Para remover tal redundância, é frequentemente utilizada uma representação poligonal. [19]

### 2.3.2 Extração de formas poligonais

A extração de formas poligonais é uma representação da *point cloud* onde é realizada uma segmentação em formas poligonais (sendo o mais comum o triângulo). Os polígonos removem a redundância da *point cloud* e focam-se apenas nos pontos necessários para obter a forma desejada. Na Figura 2-5 estão presentes representações visuais de uma extração poligonal, que demonstra a área/pontos a serem reduzidos em algo que tem significado para a cena. [19]

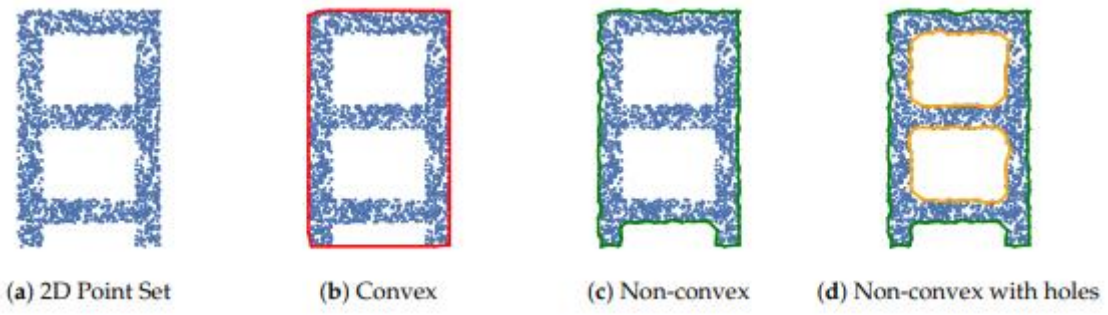


Figura 2-5 – Exemplo de uma extração poligonal sobre um plano segmentado de uma *point cloud*. (a) representação 2D dos pontos. (b) polígono convexo. (c) polígono não-convexo. (d) polígono não convexo com buracos no interior. [19]

Ao representar os pontos em forma de polígono não-convexo com buracos no interior, obtém-se as seguintes vantagens: [19]

- Reduz significativamente a memória necessária para armazenar os dados, na ordem de $n^2$;

- Maior capacidade computacional uma vez que só são tratados os polígonos de interesse;

- Ao aplicar rotinas computacionais de polígonos, é possível dilatar, corroer e simplificar os polígonos;

- Os buracos nos polígonos virtuais equivalem a buracos ou obstáculos no mundo real.

### 2.3.3 Rotulagem semântica da point cloud

Existem duas estratégias principais para a rotulagem semântica de uma nuvem:

- Associar a cada ponto ou *voxel* um rótulo relativo à cena;

- Classificar através de segmentos previamente definidos sobre a nuvem.

Para associar cada ponto a um rótulo, é necessário definir um conjunto de caraterísticas geométricas a cada ponto e relacioná-las com os seus vizinhos. [20]

*Wang et al* [21] utilizou uma *framework* de aprendizagem da rede associativa Markov (AMNs) para atribuir a cada ponto 3D uma catalogação de um tipo de classe (chão, tecto, paredes e outros objetos).

Para classificar elevados tamanhos de dados, associar cada ponto ou *voxel* fica dispendioso a de memória e computacional, sendo reduzido a classificação para segmentos da nuvem. Para classificar os segmentos, os algoritmos identificam os planos/segmentos que se intersectam, como forma de avaliar os limites dos planos e poder classificar cada plano como chão, tecto, paredes, objetos. Díaz-Vilariño et al. [22] é um exemplo sobre a classificação da *point cloud* através da obtenção das figuras geométricas do chão, paredes e objetos através da interseção entre os planos que os formam e os seus adjacentes.

### 2.3.4 Reconstrução baseada em linhas

A reconstrução baseada em linhas é uma forma de reconhecimento de linhas nas interseções existentes em planos. [20]

*Lin et al.* [23] apresentou um algoritmo de agrupamento de segmentos de linhas ao introduzir um mecanismo de deteção de falsos positivos quando era realizada a identificação das linhas numa *point cloud*. Identificou também que linhas não conectados demonstravam demasiado ruído para o caso.

Outra metodologia utilizada por *Wang* [21] foi através de *Generative Adversarial Nets (cGan)* uma abordagem de *deep learning* para obter um modelo de linhas regulares com a topologia das estruturas a representar.

*Liu* [24] necessitou de toda a *point cloud* e gerou uma rede neuronal que deteta estruturas e cria vetores gráficos das paredes e chão.

Apesar de não serem métodos diretamente relacionados com a recriação de ambientes interiores, a reconstrução baseada em linhas é eficiente na deteção dos segmentos de linha.

### 2.3.5 Reconstrução baseada em planos

O método de reconstrução baseada em planos deteta e reconstrói planos a partir da *point cloud*. Pode ser utilizado tanto na reconstrução de interiores como de exteriores. [20]

Neste tipo de reconstruções são utilizadas metodologias *model-fitting*, tais como o RANSAC e transformações de *Hough* para obter uma segmentação por planos da *point cloud*. Estes tipos de algoritmos geram planos não conectados, permitindo construir um modelo sem uma topologia espacial relacionada entre si. [25]

Estes tipos de algoritmos podem ser genericamente empregues em qualquer cena cujo ambiente tenha sido criado pelo ser humano. Deste modo, a metodologia de reconstrução por planos pode ser utilizada em grande escala, porém a utilização de recursos e custo computacional é elevado e relacionado com o tamanho da cena a reproduzir. [20]

### 2.3.6 Reconstrução baseada em volumes

A reconstrução volumétrica das cenas necessita de uma predefinição de regras. Estas regras são baseadas nas regras primitivas de divisão e fusão volumétricos.

*Oesau* [26] utilizou este tipo de metodologia para detetar os planos verticais e horizontais de paredes num ambiente 3D particionado. O modelo final de *Oesau* foi gerado através da análise de cortes gráficos sobre objetos sólidos e ocos. O método é útil para obter informação sobre o interior/exterior dos objetos, mas não da restante cena (limites, paredes, tectos, …).

Outra metodologia é a partição binária dos espaços [20, 11] pode ser utilizado para separar o espaço interior do exterior através de cortes gráficos. Apesar deste tipo de separação, este algoritmo não identifica qual é qual espaço, se o interior se o exterior.

### 2.3.7 Ball Pivoting Algorithm (BPA)

O BPA é um algoritmo que permite a reconstrução de cenas a partir de uma *point cloud*. Este algoritmo necessita do input de uma lista de pontos, uma lista de normais associadas a cada ponto, e o raio da bola que será utilizada. [27]

Tal como é possível visualizar na Figura 2-6, o algoritmo procura por três pontos que se insiram dentro de uma bola criada a partir do raio fornecido. Para esta identificação, a “bola” é colocada num ponto da nuvem fornecida, fazendo-a rodar em torno desse ponto. Ao encontrar dois outros pontos, é criado uma ligação entre os pontos, passando a bola a estar colocada num dos pontos seguintes, repetindo os passos anteriores até que seja obtida uma malha poligonal. [27]

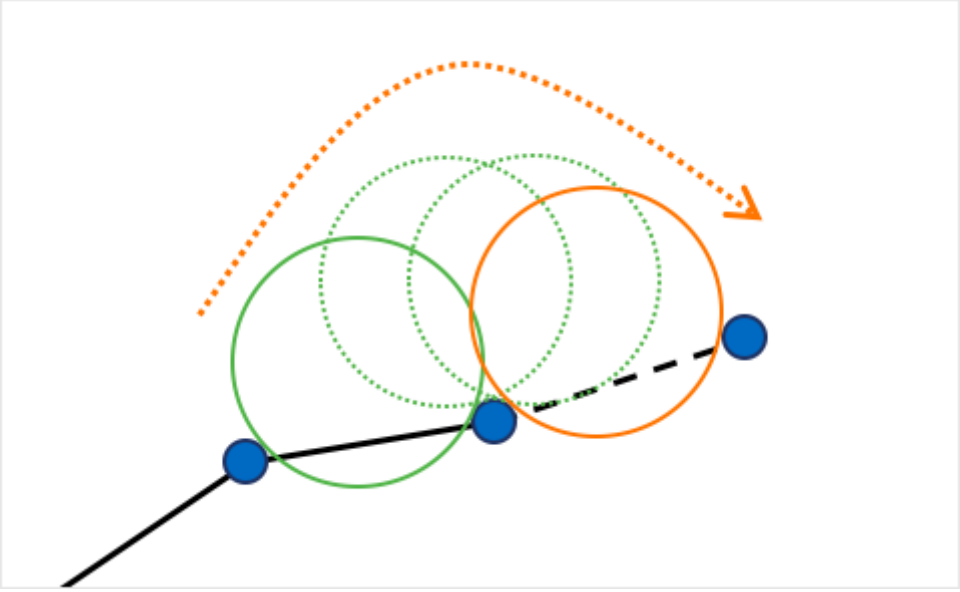

Figura 2-6 – Exemplo algoritmo de Ball Pivoting [28]

### 2.3.8 Reconstrução de Poisson

A reconstrução de *Poisson* é um algoritmo para reconstruir superfícies a partir de *point clouds*. Esta reconstrução baseia-se no problema de *Poisson* e na solução das equações provenientes do mesmo. [29]

Para a implementação deste tipo de reconstrução é necessária uma *point cloud* cujos pontos estejam orientados, ou seja, apresentem vetores normais.

Ao contrário de outras reconstruções implícitas de superfícies, que segmentam os pontos por região e tentam dar reconstruções locais aproximadas, a reconstrução de *Poisson* considera todos os pontos e gera uma solução global para a reconstrução. Esta solução é apresentada sobre a forma de uma malha poligonal. [30]

### 2.3.9 Convolutional Occupancy Network

A *Convolutional Occupancy Network* é um modelo de reconstrução apresentado por Peng *et al.* [31] que tem por base a utilização de redes neuronais convolucionais.

Este método procura caraterísticas ao nível de 3D na *point cloud* utilizando a interpolação linear. O principal objetivo é o permitir a recriação implícita das superfícies em modelos de grande escala. [31]

Para obter este objetivo, o método em questão codifica o *input (point cloud)* em grelhas 2D e 3D. Esta informação é processada pela rede convolucional que determina uma probabilidade de ocupação. [31]

## 2.4 Tecnologia existente scan

### 2.4.1 ARCore

Tecnologia da Google para a criação de experiências de realidade aumentada, permitindo aos dispositivos móveis obter informação do mundo real a partir dos sensores e câmara. Esta tecnologia é suportada por sistemas Android 7.0 (Nougat) ou superior e iOS 11.0(ou superior) e compatíveis com ARKit. [32]
O ARCore utiliza principalmente três capacidades chave, essenciais para integrar conteúdo do mundo virtual no real: [32]

- *Rastreamento* de movimentos - permite saber e localizar a posição relativa de objetos no mundo;

- Compreensão do ambiente - permite detetar o tamanho e localização das superfícies sendo elas: horizontais, verticais ou angulares;

- Estimativa da luminosidade - permite estimar as condições luminosas do ambiente.

O ARCore utiliza preocupa-se em duas coisas: rastrear a posição do dispositivo móvel à medida que se movimenta no mundo e constrói a sua própria perceção do mundo real. Para que seja possível efetuar ambos, são necessárias as capacidades referidas anteriormente, e que serão detalhadas de seguida. [32]

De forma a retirar um melhor partido das funcionalidades do ARCore, existem conceitos chave para que seja possível compreender a abordagem desta tecnologia: [33]

- **Rastreamento de movimentos** – O processo utilizado para esta finalidade é o SLAM Visual. Tal como referido anteriormente, este processo seleciona pontos chave que permitem calcular distâncias no mundo virtual;

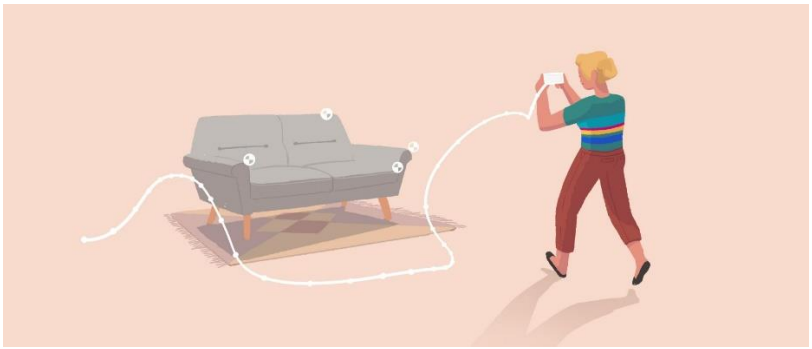

Figura 2-7 – Representação do SLAM [33]

- **Compreensão da profundidade –** O ARCore possibilita a criação de mapas e imagens de profundidade. Esta informação pode ser útil para criar colisões entre objetos e tornar o mundo virtual mais realista através da perceção de profundidade/ afastamento dos objetos;

- **Compreensão do ambiente -** A compreensão do meio que rodeia o dispositivo baseia-se nos pontos caraterísticos adquiridos ao longo do tempo, e da criação de clusters desses mesmo pontos, formando assim planos para as superfícies (2.3.1).

- **Interação com o utilizador –** O utilizador pode selecionar/interagir com planos, objetos ou pontos virtuais através da interação com o ecrã do dispositivo;

- **Estimativa luminosa** – Ao utilizar a luminosidade de cada objeto da cena é obtido um maior realismo. Esta tecnologia deteta essa informação luminosa, capturando uma média de intensidade luminosa e a correção da gama colorida do objeto.

- **Ancoras e rastreadores –** A pose e localização do utilizador pode variar ao longo do tempo. Para garantir a persistência da informação ao longo da experiência do utilizador, são utilizadas ancoras e rastreadores dos objetos em questão.

### 2.4.2 ARKit

O ARKit é uma tecnologia que cria experiências de realidade aumentada a partir das potencialidades do iOS. Utiliza o sensor de movimentos, a captura de imagem, e a capacidade de processamento e as conveniências de display dos dispositivos. [34]
O ARKit está disponível para iOS com processadores A9 ou superiores.
O ARKit permite:

- **Rastreamento** – Identifica e localiza o dispositivo na cena, a partir de dados do sensor de movimentos e imagens da câmara. Para localizar o dispositivo na cena e a localização do mesmo, é ainda utilizada a odometria inercial visual. A odometria inercial visual é um processo robótico que analisa as imagens associadas à cena e define, a partir delas, uma posição e orientação do dispositivo;

- **Compreensão da cena –** Permite uma compreensão do ambiente em torno do dispositivo, a partir da deteção de planos existentes e de atributos como a luminosidade e interseção de planos.

- **Renderização** – É utilizada a informação proveniente das imagens capturadas pela câmara e informação dos sensores de movimento, para realizar a recriação de uma cena visualmente equiparável ao real;

- **Luminosidade -** A câmara do dispositivo é utilizada pelo ARKit para avaliar e estimar o valor de luminosidade existente na cena e em cada objeto nela presente, traduzindo esses valores nos objetos virtuais aquando da sua renderização;

- **Ancoras –** Ao colocar objetos na cena virtual, são utilizadas ancoras para guardar a informação sobre a sua posição e orientação no mundo.

### 2.4.3 LiDAR

LiDAR é um método utilizado por sensores remotos através do uso de lasers para medir distâncias a terrenos. Estes pulsos de laser quando combinados com outras fontes de dados geram informação precisa acerca das caraterísticas e forma das superfícies.

O LiDAR consiste principalmente num scanner, um laser e uma forma especializada de GPS. Este serve para mapear as redondezas do sensor com precisão e flexibilidade. [35]

Existem dois principais sistemas LiDAR: aéreos e terrestres. Os Lidar aéreos emitem um feixe luminoso para a superfície de modo a calcular as distâncias entre o dispositivo e o chão. Os Lidar aéreos são principalmente: [36, 37]

- Topográficos – usados para derivas modelos da superfície;
- Batimétricos – utilizados para obter a elevação do fundo oceânico e profundidade do mar. Pode também ser utilizado para procuras marítimas.

Os LiDAR terrestres são utilizados para obter informação precisa à superfície. Este tipo de informação serve principalmente para recriação de espaços exteriores ou gerar *point clouds* dos edifícios. Este tipo de LiDAR está dividido em dois tipos: [36, 37]

- Móveis – colecionam informação em forma de pontos cartesianos, a partir de uma plataforma móvel. Utilizados na criação e análise de locais. Composto pelo sensor LiDAR, câmaras, GPS e um sistema de navegação inercial.
- Estáticos – colecionam informação a partir de uma plataforma estática. Consiste apenas num sensor e sistema de imagem.

### 2.4.4 Matterport

Matterport é um standard para as capturas espaciais 3D, sendo também uma plataforma que permite capturar e tornar os espaços reais num mundo virtual imersivo. [38]

Para realizar scans com o Matterport é necessária uma câmara compatível (câmaras 360, iPhone/iPad Pro, Matterport câmara) e uma subscrição na plataforma. [38]

Após a realização do scan, o Matterport contém componentes de inteligência artificial que identifica os vários objetos e espaços 3D e reconstrói-os numa visita virtual. Tem também capacidade de tornar imagens 360º em espaços virtualizados. [38]

### 2.4.5 Polycam

O *Polycam* é uma aplicação para iOS que utiliza as funcionalidades disponibilizadas com a introdução de sensores LiDAR nos iPhones. Esta aplicação permite o scan e recriação dos

espaços para o telemóvel, sem ser necessário um perito na área para realizar esta digitalização. [39]

### 2.4.6 Canvas.io

O canvas.io permite ao utilizador usufruir do LiDAR incorporado nos iPhones para efetuar scans de habitações, scans que são recriados em modelos CAD. Os modelos recriados podem ser utilizados para verificar a planta das habitações bem como efetuar medições em tempo real sobre o modelo. [40]

Segundo a *Occipital*, o canvas.io serve de suporte para realizar funções de arquitetura e de design dos interiores das habitações.

### 2.4.7 Comparação de tecnologias scan

Quanto às tecnologias de scan analisadas, o ARCore e ARKit são semelhantes nas suas funcionalidades e são de utilização gratuita. O *Matterport* e o *Canvas.io* necessitam de investimento numa subscrição para usufruir das suas funcionalidades e o *Polycam* permite realizar os scans 3D de uma forma gratuita, mas para utilizar os scans fora da aplicação é necessário um plano *premium*.

Para uma utilização mais generalizada, ao nível de scan, o ARCore é o que permite uma maior abrangência de plataformas, uma vez que é possível utilizar em Android e iOs, enquanto os restantes são apenas em iOS.

Quanto à realização de análises em cima do modelo 3D (como por exemplo, permitir obter medições de distâncias) os únicos que o permitem efetuar é o Canvas.io e o Matterport.

É possível analisar estas comparações de uma forma mais visual na Tabela 2-1.

Tabela 2-1 – Comparação de tecnologias scan

| | Pago | Android | iOS | Odometria | Processamento *Point Cloud* | Processamento em malha poligonal | Scan 3D | Web Export | Análise Modelo 3D |
|---|---|---|---|---|---|---|---|---|---|
| **ARCore** | - | X | X | X | X | - | X | - | - |
| **ARKit** | - | - | X | X | X | - | X | - | - |
| **Matterport** | X | - | X | X | X | X | X | X | X |
| **Polycam** | X | - | X | X | - | X | X | X | - |
| **Canvas.io** | X | - | X | X | X | X | X | X | X |

## 2.5 Tecnologias Modelação

### 2.5.1 Point Cloud Library (PCL)

A PCL é uma biblioteca *open source* que contém mecanismos de manipulação e processamento de *point clouds* e geometria em três dimensões. Está integrada no ROS (*Robot Operating System*), tendo sido utilizada em vários projetos na área de robótica. [41]

Esta biblioteca foi desenvolvida em e para C++, tendo como pontos fulcrais a performance e a eficiência nos CPU no processamento dos dados. Contém vários tipos de algoritmos disponíveis para que seja possível incorporar a multitude da informação 3D obtida pelo scan do mundo real. [41]

### 2.5.2 Open3D

O Open3D é uma plataforma open source que suporta um desenvolvimento rápido de software que visa trabalhar com dados 3D. Esta biblioteca disponibiliza algoritmos em C++ e Python para manipulação e tratamento dos dados provenientes de sensores 3D, providenciando ainda estruturas para armazenar tal informação. [42]

É uma biblioteca otimizada e que suporta paralelismo, tendo como principais funcionalidades:

- Estruturas 3D de dados;
- Algoritmos de processamento de dados 3D;
- Reconstrução de cenas;
- Alinhamento de superfícies;
- Visualização 3D;
- Renderização baseada em físicas,
- Suporte de PyTorch e TensorFlow para criação de algoritmos 3D de *machine learning*;
- Aceleração de GPU para operações 3D nucleares.

### 2.5.3 TorchPoints3D

O *TorchPoints3D* é uma *framework* em *Python* de análise de *point clouds*. Para a realização desta análise, esta *framework* disponibiliza implementações de modelos de *deep learning* para utilização em *point clouds.* [43]

Desta forma, esta tecnologia visa reduzir o esforço necessário para implementar os modelos de *deep learning* necessários para reconstruir a *point cloud*, e permite a reutilização destes mesmos modelos. [43]

Um ponto importante é a base desta framework assentar em formas de *deep learning* identificadas como *state-of-the-art*.

### 2.5.4 Simplygon

O Simplygon é um *SDK* (*Software Development Kit)* que providencia algoritmos de otimização para artefactos 3D. Este *software* pode ser incorporado nas pipelines de desenvolvimento nativamente utilizando o SDK do *Simplygon* ou através das API disponibilizadas em *Python, C++* e *C#.* [44]

Apesar de ser uma ferramenta desenvolvida a pensar num ambiente de desenvolvimento de jogos, esta pode ser utilizado para outro tipo de situações, devido à sua flexibilidade e integrabilidade em script e ferramentas de desenvolvimento que utilizem *Python e C#*. [44]

### 2.5.5 ThreeJs

ThreeJs é uma biblioteca 3D que facilita a incorporação de conteúdos 3D numa página *web*. Utiliza o *WebGL* para desenhar cenas 3D, porém não é apenas *WebGL*, uma vez que para desenhar algo em *ThreeJS* é necessário muito menos código do que em *WebGL*.

É uma biblioteca para JavaScript que se encarrega de partes da cena como a iluminação, sombras, materiais, texturas, matemática 3D que por outra via teria de ser codificado pelo programador. [45]

### 2.5.6 Comparação de tecnologias criação modelos 3D

Quanto às tecnologias de criação de modelos 3D, o Open3D é o mais versátil pois permite a sua utilização em C++ e Python, enquanto o PCL apenas o permite em C++ e o TorchPoints3D em Python.

O PCL e o Open3D têm conectores que permitem facilitar a integração com tecnologias de scan 3D.

O PCL, Open3D e TorchPoints3D permitem processar as *point clouds* e recriar as mesmas em modelos 3D.

Tanto o Open3D e o Simplygon podem ser utilizados para simplificar os modelos 3D obtidos.

O Open3D e o PCL permitem visualizar os modelos 3D gerados na altura das suas criações enquanto os restantes não o permitem.

O Simplygon apesar de ser um plugin para interfaces gráficas de modelação 3D (Como o *Blender ou Maya*), disponibiliza APIs para a sua utilização *Standalone* através de *Python, C++ ou C#*.

Na Tabela 2-2 estão presentes as tecnologias e respetiva comparação entre as mesmas.

Tabela 2-2 – Comparação tecnologias criação modelos 3D

| | **Open-Source** | **Python** | **C++** | **Scan 3D** | **Processamento *Point Cloud*** | **Processamento Malha Poligonal** | **Simplificação Modelos 3D** | **Visualização Modelos 3D** |
|---|---|---|---|---|---|---|---|---|
| **PCL** | X | - | X | X | X | X | - | X |
| **Open3D** | X | X | X | X | X | X | X | X |
| **TorchPoints3D** | - | X | - | - | X | X | - | - |
| **Simplygon** | - | X | X | - | - | X | X | X |

# 3 Análise de negócio

No presente capítulo é apresentada a análise de negócio, incluindo detalhes sobre o contexto do problema, processos e intervenientes, simplificação e generalização, bem como as restrições encontradas.

## 3.1 Detalhes sobre o contexto e problema

Com o avanço tecnológico, surgiu a necessidade de permitir uma experiência mais imersiva aos clientes que visam a compra de imóveis. Para corresponder a tal necessidade, a criação de uma visita virtual aos imóveis foi a solução encontrada.

Para que esta visita virtual seja possível de realizar é necessário que o vendedor do imóvel realize um scan ao interior da habitação, seguido de um processamento desse scan pelo servidor e por fim a criação de uma cena virtual que seja representada no site onde fica disponível para os possíveis compradores.

Deste modo, é possível dividir este problema em três partes:

- Realização do scan – o vendedor executa o scan da habitação e os dados resultantes desse scan são enviados para o servidor;
- Processamento dos dados obtidos pelo scan – após receber a informação do scan, o servidor cria um modelo 3D a partir da informação recolhida, que é posteriormente enviada para uma base de dados;
- Recriação numa cena virtual – construção de uma visita virtual a partir do modelo 3D criado através do processamento, sendo depois renderizado numa página web, dando assim forma do possível comprador de visualizar a habitação.

## 3.2 Processos e intervenientes

Nesta subsecção são apresentados o modelo de domínio, as vistas de cenário, vistas lógicas e vistas de processos como forma de permitir uma melhor perceção acerca dos processos e intervenientes presentes no projeto apresentado neste documento.

### 3.2.1 Modelo de domínio

Nesta subsecção é apresentado o modelo conceptual do problema, Figura 3-1, sendo especificados os conceitos do negócio.

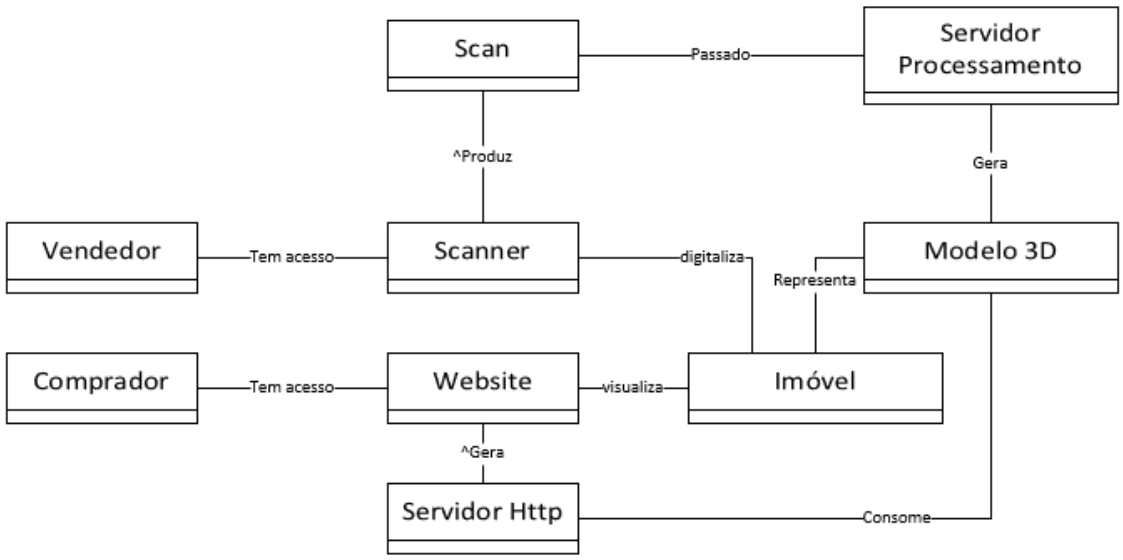


Figura 3-1 – Modelo Conceptual

### 3.2.2 Vista cenários

A vista de cenários do sistema descrito neste documento corresponde ao diagrama de casos de uso representado na Figura 3-2, intervindo dois autores:

- Vendedor – Realizar scan do imóvel
- Comprador – Visualizar imóveis
- Servidor – Processar imóvel para modelo 3D

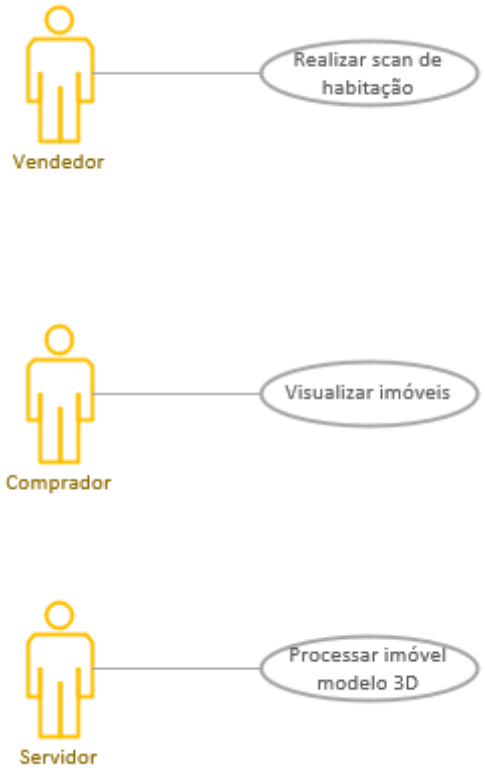


Figura 3-2 – Diagrama de casos de uso

### 3.2.3 Vista lógica

No diagrama de componentes da Figura 3-3, podem ser observadas as interações entre os atores, como forma de realizarem as suas tarefas, existindo 4 componentes:

- Scanner - dispositivo de entrada que permite a digitalização das habitações;
- Servidor HTTP visualização – servidor que disponibiliza o website onde o comprador irás ver o imóvel e onde é feita a renderização do modelo 3D;
- Servidor HTTP introdução – servidor que recebe e armazena os dados obtidos pelo scanner em persistência.
- Servidor processamento de imagens – servidor que efetua o processamento dos dados obtidos pelo scanner para um modelo 3D;
- Persistência – componente que permite armazenar os dados do scanner e modelos 3D para uso posterior.

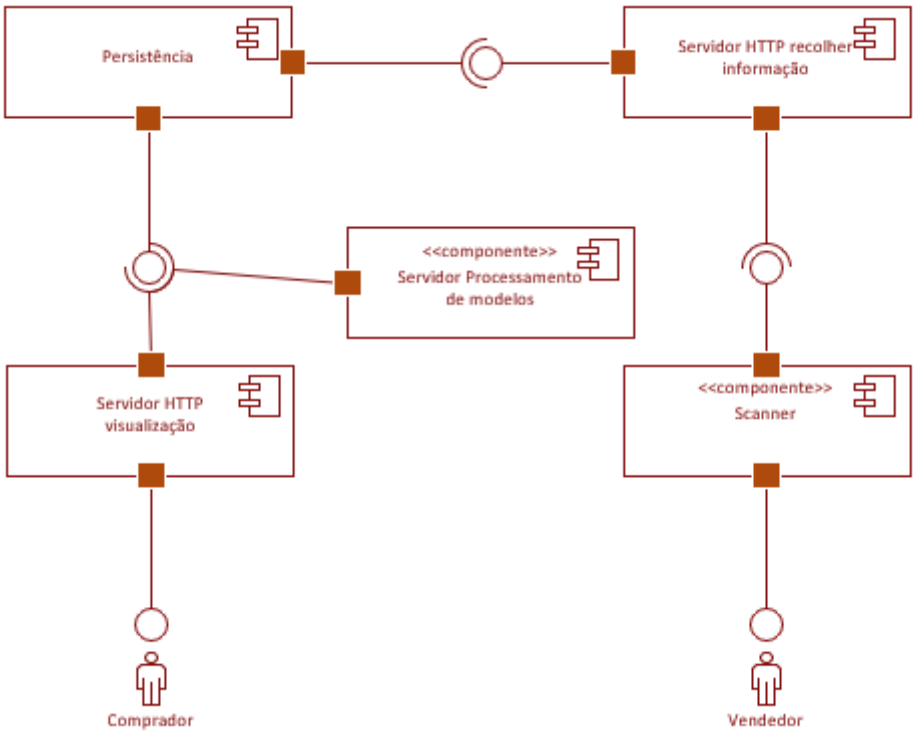


Figura 3-3 - Diagrama componentes alto nível

### 3.2.4 Vista processos

De seguida são apresentadas as vistas de processos sobre a forma de diagrama de sequência de casos de uso da solução.

#### 3.2.4.1 Realizar scan da habitação

Este caso de uso tem os passos da Figura 3-4:

- O Vendedor inicia o processo de scan da habitação;
- O módulo de Scan efetua a digitalização do imóvel;
- O Scanner retorna os dados relativos à digitalização do imóvel;
- Os dados obtidos são guardados na base de dados e retorna à informação de sucesso da operação

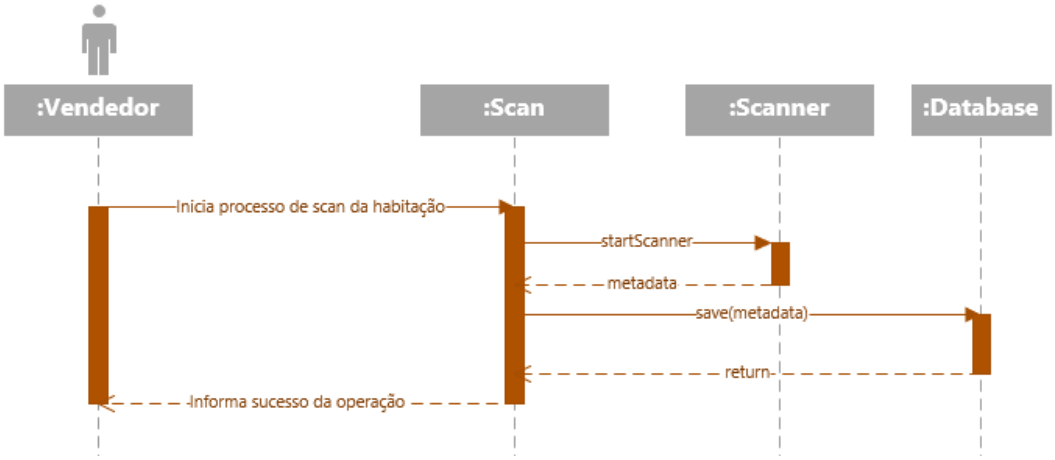


Figura 3-4 - Diagrama sequência: Realizar scan da habitação

#### 3.2.4.2 Processar imóvel em modelo 3D

Este caso de uso tem os passos da Figura 3-5:

- Quando é inserida um novo scan da habitação é despoletado o caso de uso;
- A base de dados retorna o resultado;
- É realizado o processamento do scan para um modelo 3D;
- O modelo gerado é armazenado em persistência.

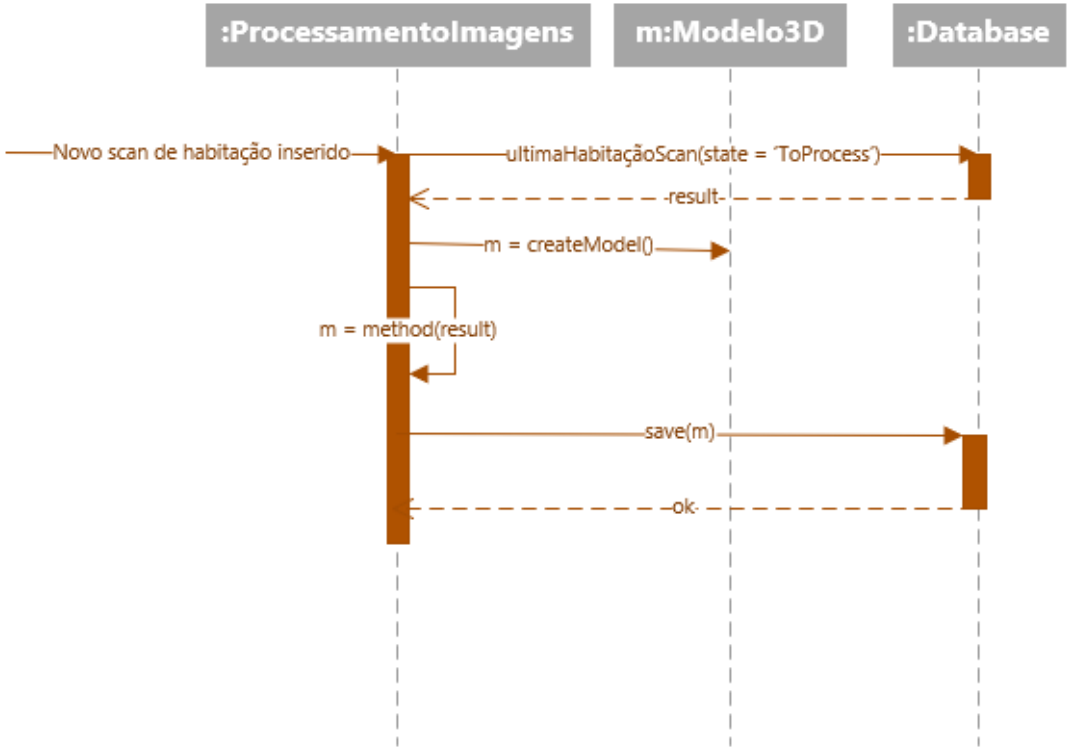


Figura 3-5 – Diagrama sequência: Processar imóvel em modelo 3D

#### 3.2.4.3 Visualizar imóveis

Este caso de uso tem os passos da Figura 3-6:

- O Comprador inicia o processo de visualizar o imóvel 3D;
- O Servidor obtém a informação do modelo 3D do imóvel, existente na persistência;
- O Servidor cria a cena virtual e recria nela o modelo 3D do imóvel;
- O Servidor mostra a cena ao utilizador.

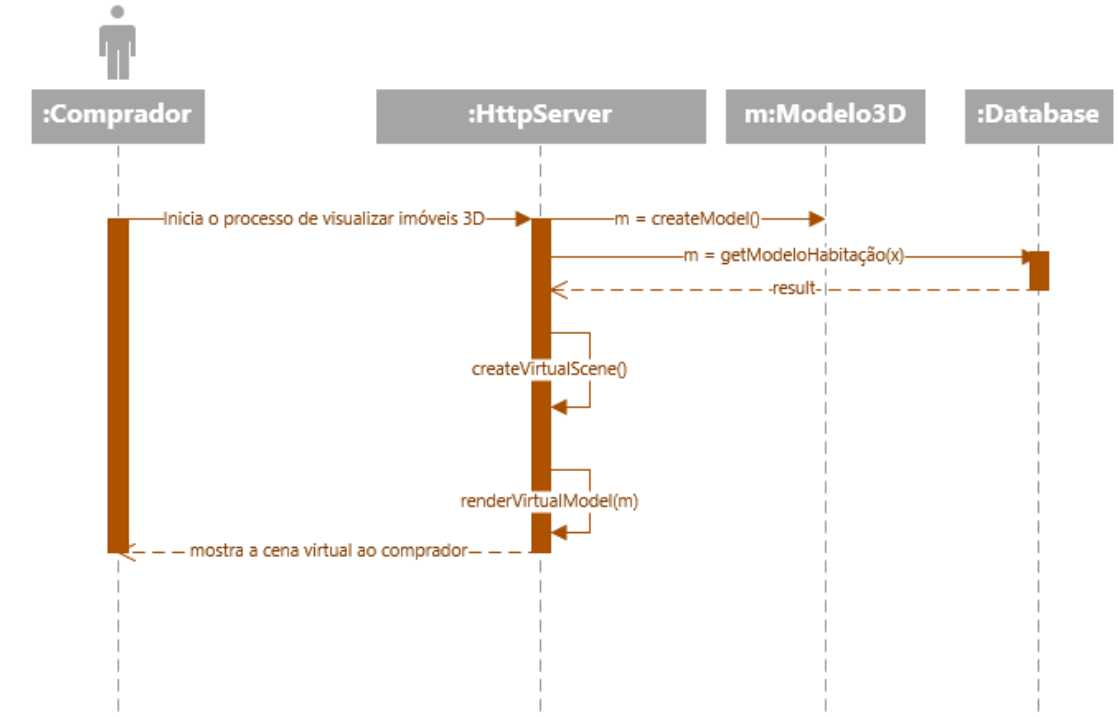


Figura 3-6 – Diagrama sequência: Visualizar imóveis

## 3.3 Restrições existentes

Nesta secção são apresentadas as restrições existentes para o desenvolvimento da solução apresentada no documento.

Para a criação da solução proposta, é necessário que:

- As exceções sejam tratadas;
- Deve ser possível alterar os recursos e tecnologia utilizada entre componentes;
- Deve ser garantida a futura manutenibilidade e atualização do sistema desenvolvido.

# 4 Análise de valor

Tendo por base o contexto apresentado anteriormente, o presente capítulo tem como finalidade realizar a análise de valor do projeto apresentado por este documento.

A Análise de Valor (VA) é considerada como um processo formal, sistemático e organizado de análise e avaliação. É um processo de gestão que requer planeamento, controlo e coordenação. [46]

O processo de VA está dividido em seis fases, tendo cada uma delas as suas técnicas específicas, como demonstra Figura 4-1 :

1. Orientação;
2. Identificação funcional;
3. Alternativas criativas;
4. Análise e avaliação;
5. Implementação.

É necessário ter em conta que o cliente é quem cria valor para o negócio, logo a análise de valor tem de equacionar a utilidade e o desejo de compra do cliente, para que o produto obtenha um maior valor para a empresa. [46] [47]

Tendo em conta a descrição efetuada da análise de valor e dos seus processos, de seguida é efetuada uma análise ao problema/solução descrito neste documento através de: Modelo de desenvolvimento de novo conceito (NCD), Proposta de valor e FAST.

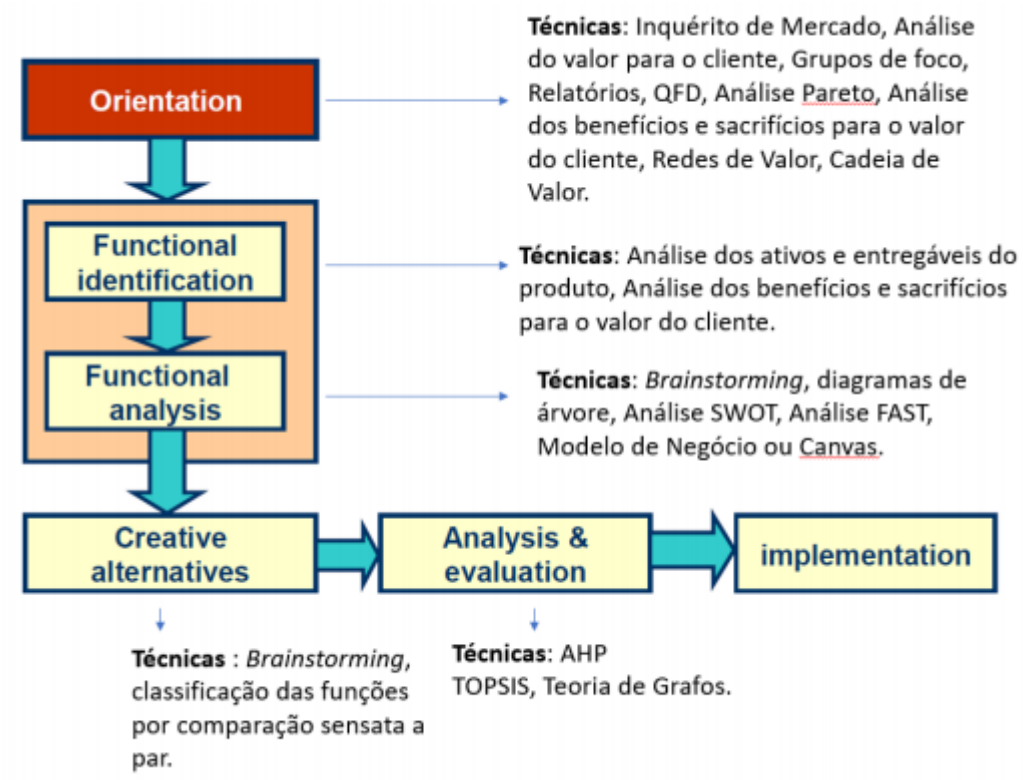


Figura 4-1 – Análise de Valor: Processos e técnicas
Fonte: [46] (adaptado para português)

## 4.1 Modelo de desenvolvimento de novo conceito (NCD)

Foi encontrada uma oportunidade de inovação da tecnologia existente, de modo a aumentar o valor do produto já fornecido pela organização. Para identificar esta oportunidade, será utilizado o 'modelo de desenvolvimento de novo conceito' que fornece uma linguagem e perceção mais comum. [48]

Este modelo tem como foco o progresso do produto e é dividido em três partes (como demonstra a Figura 4-2) [49]:

- **Motor –** parte central que contém os elementos auxiliadores do processo. Estes elementos correspondem a fatores organizacionais como a gestão envolvida, visão, estratégia, recursos, cultura e colaboração;
- **Roda –** parte interior do modelo constituída por cinco atividades:
    - *Identificação da oportunidade* – Esta fator é uma escolha a seguir por parte da empresa, sendo que a sua essência se foca na origem e métodos para identificar as oportunidades: mapeamento, análise de tendências de mercado e tecnológicas, verificação de clientes e competição.

- Análise da oportunidade – Neste elemento é confirmada a oportunidade encontrada anteriormente. Nesta fase é procurada informação adicional sobre a oportunidade e as tecnologias a utilizar.
- *Produção e enriquecimento da ideia* – Surge uma ideia concreta e maturada do que se pretende realizar. Pode envolver clientes, novas soluções tecnológicas, variedade no incentivo e estímulos e avaliações às necessidades do mercado.
- *Seleção de ideia* – É necessário realizar uma boa seleção das ideias a implementar, porém não existe algo padronizado para esta seleção, dependendo de cada organização.
- *Definição de conceito* – É o produto final do processo. Deve ser apelativo ao investimento na proposta.

- **Aro –** parte externa, representativa dos fatores externos à organização e que influenciam o motor e as atividades a executar. São normalmente questões: políticas, económicas, sociais, tecnológicas e legais.

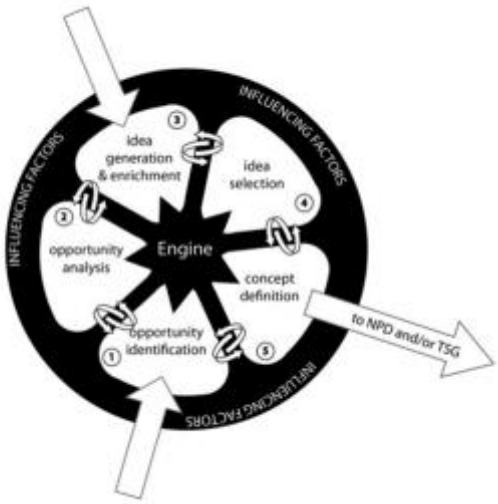


Figura 4-2 – Modelo de desenvolvimento de novo conceito [49]

Este é um modelo circular, significando assim um fluxo de ideias entre os cinco elementos. As setas e numerações indicam os pontos iniciais para o início do projeto. [49]

Deste modo, e tendo como base o modelo referido anteriormente, foram identificados os seguintes elementos no contexto do presente documento:

- **Identificação da oportunidade –** O sistema de apresentação de interiores de habitações aos clientes, presente na organização, passou por três fases anteriores:

1. Apresentação de uma ou várias imagens – nesta fase o cliente inseria fotografias da habitação, fotografias que eram apresentadas no website de venda de imóveis.

2. Apresentação de uma vista 360º do interior da habitação - nesta iteração o cliente introduzia uma fotografia panorâmica das secções da habitação, fotografia que era processada e colocada no website.

3. Apresentação de uma vista 360º da habitação com possibilidade de deslocação entre secções – nesta iteração foi realizada uma melhoria em relação à iteração anterior, que consistia na possibilidade do comprador se deslocar de uma forma mais acessível entre as divisões da habitação.

No sistema atual vigora a última iteração apresentada, mas surgiu a oportunidade de apresentar a habitação ao utilizador de uma forma mais imersiva, devido à evolução nas áreas de realidade virtual/aumentada e da digitalização dos espaços.

Deste modo, a oportunidade identificada pela organização visa a utilização deste tipo de tecnologias para criar um ambiente 3D que permita ao comprador de imóveis visualizar de uma forma mais imersiva o interior da habitação.

- **Análise da oportunidade –** Tendo em conta os aspetos referidos anteriormente na identificação da oportunidade, foi reconhecida a possibilidade de acréscimo de valor da tecnologia, por parte da organização, caso a referida inovação fosse implementada.

- **Produção e enriquecimento da ideia –** Como forma de enriquecimento da ideia, numa primeira fase foram reunidos conceitos referentes ao scan de locais, tratamento e visualização de dados 3D. De seguida, foi realizado um levantamento do estado da arte de tecnologias relevantes para o problema.

- **Seleção da ideia –** Foi efetuada uma análise entre as funcionalidades e limites das soluções existentes, bem como hardware já presente na organização, tendo sido efetuada a escolha de utilizar o Open3D para processar os dados, o *Azure Kinect* ou iOS para efetuar scans às habitações.

- **Definição de conceito –** O objetivo do projeto é de proporcionar aos compradores de imóveis uma nova forma de visualização dos interiores das habitações.

## 4.2 Proposta de valor

Para avaliar e obter o valor possível da proposta, é necessário avaliar o que a organização pode oferecer, quais os alvos das propostas, como a realizar e os ganhos de realizar tais propostas. Deste modo, são respondidas as perguntas clássicas de “o quê”, “quem”, “como” e “quanto”, o que permite criar uma proposta de valor sólida. [50]

Deste modo, a proposta de valor obtida para este documento é o facto da necessidade de inovar o modo como são apresentados os imóveis aos clientes/compradores, criando ambientes mais imersivos através do scan e recriação de cenários virtuais, aumentando o detalhe e percetividade do cliente, aumentando a procura e favorecendo a experiência do utilizador.

## 4.3 FAST

O diagrama FAST é um diagrama lógico de funcionalidades. Este diagrama retrata relações entre diversas funcionalidades de um sistema, sendo assim um diagrama orientado por funções e não por tempo. Tem como finalidade a disposição de uma lista de funcionalidades de uma forma lógica e percetível à compreensão por terceiros. [51]

Este diagrama tem no seu núcleo um caminho crítico, regido pelas perguntas “Como? /Porquê?”, tendo como resposta a funcionalidade à sua direita (no caso do Como?) ou à sua esquerda (no caso do Porquê?). [51]

Assim sendo, para a solução descrita neste documento, foram encontradas as seguintes funcionalidades:

- Vender casa;
- Comprar casa;
- Visualizar imóvel;
- Criar cena 3D habitação;
- Obter modelo 3D habitação;
- Processar informação 3D do imóvel;
- Obter dados Scan;
- Realizar scan habitação;
- Aceder ao browser;
- Aceder à internet;
- Melhorar perceção imóvel;
- Melhorar performance cena;
- Melhorar performance modelo 3D;
- Guardar modelo 3D;
- Guardar dados do scan;
- Utilizar câmara.

Após a identificação das funcionalidades, foram identificadas as funcionalidades que seriam críticas para a solução, criando assim um caminho principal para o diagrama FAST. Estas funcionalidades são: Vender casa; comprar casa; visualizar imóvel; criar cena 3D habitação; obter modelo 3D da habitação; processar informação 3D do imóvel; obter dados do scan; realizar scan da habitação; utilizar câmara.

De seguida, foram associadas as restantes funcionalidades auxiliares às funcionalidades críticas, dando origem ao diagrama FAST apresentado em seguida. (Figura 4-3 - Diagrama Fast)

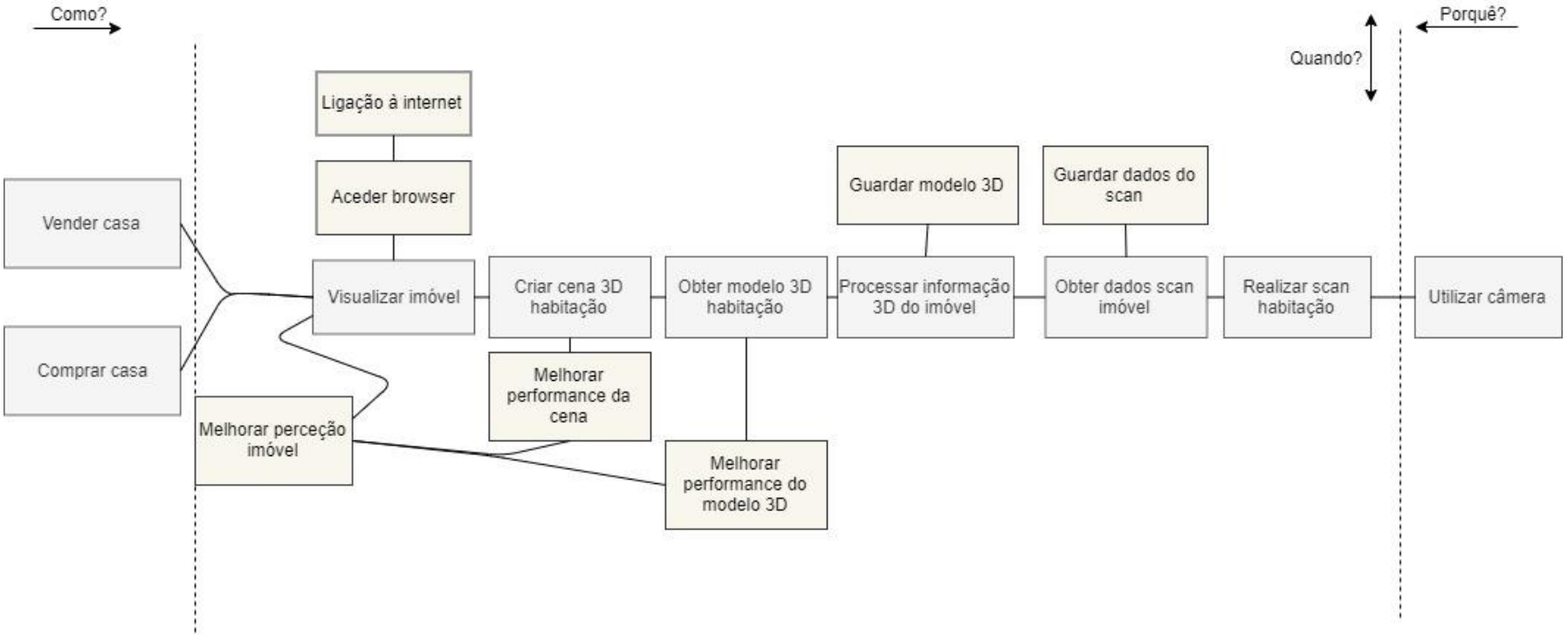


Figura 4-3 - Diagrama Fast

# 5 Desenho da Solução

Este capítulo tem como foco a análise e desenho da solução desenvolvida segundo as existentes possibilidades.

Deste modo numa primeira fase foi feita a análise a possíveis abordagens tecnológicas para desenvolver a solução.

Em seguida é apresentado o desenho arquitetural para a solução proposta, tendo base nas tecnologias escolhidas.

## 5.1 Abordagens alternativas

Nesta secção serão apresentadas diversas alternativas para a criação dos modelos para recriação dos espaços. Estas abordagens serão analisadas através do método AHP.

O método *Analytic Hierarchy Process* (AHP) foi um método desenvolvido com o intuito de apoiar decisões através do ranking ou estruturação da informação, tendo por base critérios bem definidos, que são necessário para a solução. [52] O método AHP tem como objetivo dividir os problemas de decisão em níveis hierárquicos como forma de facilitar a compreensão e avaliação do problema. [47]

A implementação da metodologia AHP é composta por 7 fases: [47]

1. Construção da árvore hierárquica;
2. Comparação dos critérios e alternativas;
3. Prioridade relativa de cada um dos critérios;
4. Avaliação da consistência das propriedades relativas encontradas;
5. Construção da matriz de comparação;

6. Obtenção da prioridade de cada uma das alternativas;
7. Decisão sobre a alternativa.

Cada uma das fases do método AHP são apresentadas no Anexo – Método AHP de forma mais pormenorizada.

As alternativas a serem testadas pelo método AHP são a Matterport, a Point Cloud Library (PCL)e o Open3D.

Os critérios que serão analisados são: o custo de utilização, escalabilidade, facilidade de implantação na nuvem, versatilidade da tecnologia quanto a métodos de recriação de ambientes.

Sendo que não existe informação clara e quantitativa do desempenho de cada uma das tecnologias, este critério não pode ser tido em conta para a análise. Deste modo foi obtida a árvore hierárquica de decisão da Figura 5-1 com:

- Objetivo – Definir a tecnologia a utilizar;
- Alternativas – Matterport, PCL e Open3D;
- Critérios – Custo, Escalabilidade, Facilidade de implantar na nuvem, Versatilidade.

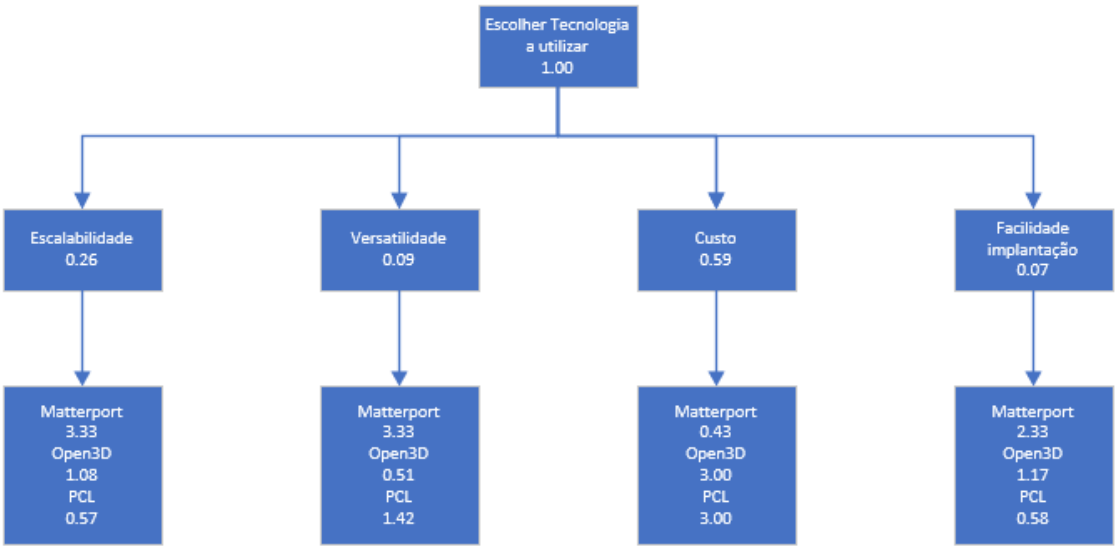


Figura 5-1 – Árvore hierárquica de decisão

Após realizada a análise dos critérios e das tecnologias em questão, através do método AHP, foi possível concluir que a tecnologia mais adequada para a resolução do problema é o Open3D.

A *Matterport*, apesar de apresentar melhores resultados em três dos quatro critérios em análise, uma vez que o custo é um fator importante para a decisão da tecnologia, esta ficou afetada devido ao elevado custo que é necessário suportar para utilizar este tipo de

tecnologia. Assim sendo, foi decidida a utilização de uma tecnologia open-source, sendo que o *Open3D* se destacou em relação ao *Point Cloud Library (PCL).*

## 5.2 Desenho arquitetural

O desenho arquitetural tem como objetivo descrever o sistema proposto como solução para o problema.

Para representar o desenho arquitetural planeado para a solução foi aplicada a abordagem de Kruchten [53], que refere o desenho arquitetural como um modelo de 4+1 vistas, demonstrando assim várias perspetivas referentes ao sistema.

- A **vista lógica** pretende descrever os requisitos funcionais do sistema. Decompõe os sistemas num conjunto de abstrações e elementos estruturais. [53, 54] Neste caso, serão utilizados diagramas de componentes para descrever a vista lógica do sistema;

- A **vista de processos** tem por foco requisitos não funcionais, tal como performance, fluxo de informação, paralelismo de tarefas entre outros aspetos não funcionais. [54] Esta vista tem ainda como objetivo demonstrar as colaborações entre os diversos elementos. [53] No presente caso esta vista será apresentada sob a forma de diagramas de sequência.

- A **vista de implementação** tem por foco a organização e gestão do desenvolvimento aplicacional. [54] Permite ainda descrever as dependências e o acoplamento entre dos diversos packages do software existente. [53]

- Vista física do sistema (**vista de implantação**), que tem como principais componentes as infraestruturas e forma de implantação utilizadas pela solução. Esta vista é descrita através dos diversos nós de processamento e a rede física que interliga os mesmos. [53, 54]

- **Vista de cenários** é uma vista adicional, que tem por objetivo demonstrar alguns cenários importantes do sistema como forma de comprovar a interligação entre as 4 vistas anteriores. [53] Esta vista permite ainda descrever os casos de uso mais importantes e os seus requisitos não funcionais.

De seguida é apresentado o desenho arquitetural do sistema através do modelo de vistas 4+1 de Kruchten [53], representado quatro vistas: lógica, processos, implantação e cenários.

### 5.2.1 Desenho arquitetural da solução

O desenho da solução proposto é baseado no modelo de vistas 4+1 de Kruchten e, tal como referido anteriormente (5.2), este desenho irá ter por base as vistas: lógica, processos, implantação e cenários.

#### 5.2.1.1 Vista lógica

Para representar a vista lógica da solução encontrada terá por base o diagrama da Figura 3-3 que apresentava de uma forma mais genérica a composição do sistema. Na Figura 5-2 representa a composição do sistema nas várias componentes, bem como a interação entre si:

- Scanner - dispositivo de entrada que permite a digitalização das habitações;
- Visualização browser - browser que disponibiliza o imóvel onde o cliente pode ver o imóvel e onde é feita a renderização do modelo 3D.
- Open3D – realiza o processamento dos dados obtidos pelo scanner, tornando as *point clouds* em modelos 3D, passíveis de ser renderizados num browser;
- *Storage Point Clouds* – componente de persistência que permite armazenar os dados do scanner obtidos do scanner para uso posterior;
- *Storage* Modelos 3D – componente de persistência que armazena os modelos 3D para serem utilizados para visualização;

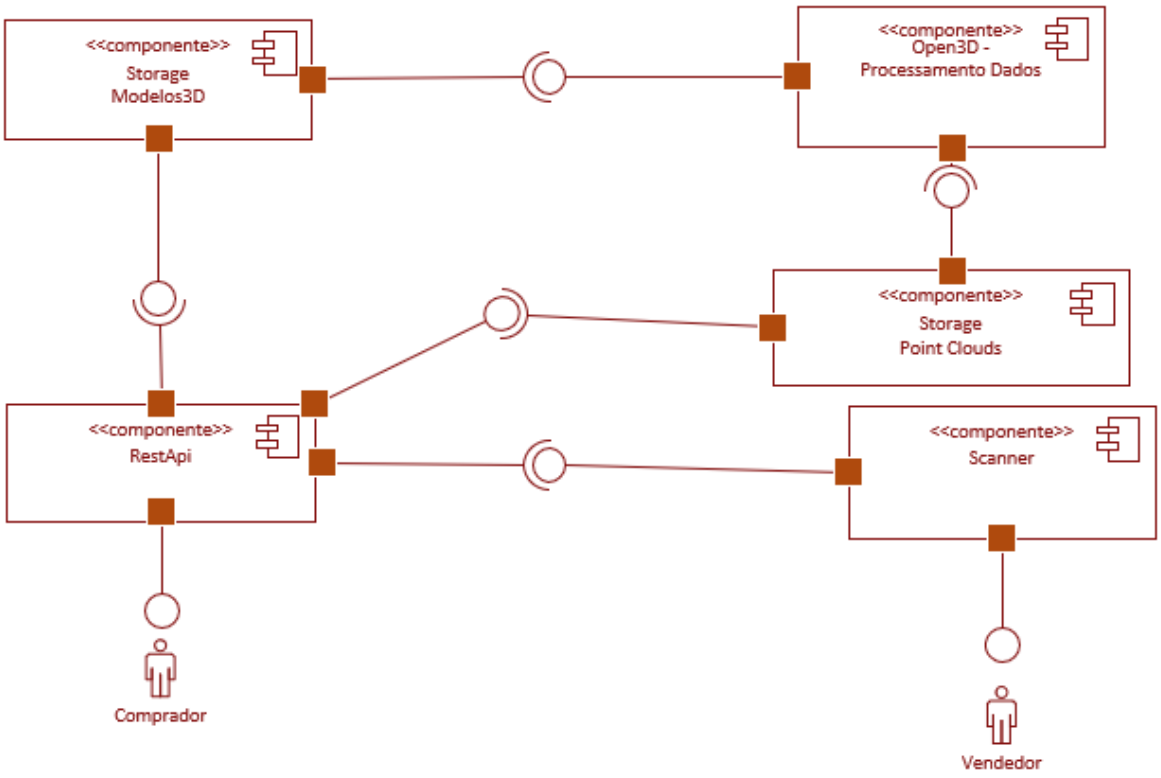


Figura 5-2 – Vista lógica: Diagrama componentes

#### 5.2.1.2 Vista de processos

Na vista de processos são evidenciados os casos de uso principais para o sistema. Tal como referido anteriormente na Análise de negócio quando foram apresentados os processos do sistema (Figura 3-2), foram identificados três processos principais.

##### 5.2.1.2.1 Processar informação dos imóveis em modelos 3D

Este processo é desencadeado quando existe a inserção de uma nova *point cloud* para ser processada. O processo tem o seguinte progresso:

- Quando é inserida um novo scan da habitação na *storage* é despoletado o caso de uso;
- A função obtém a informação inserida na *storage*;
- A informação é processada para um formato de modelo 3D;
- O modelo gerado é armazenado em *storage*.

No diagrama abaixo é possível visualizar o desenrolar do procedimento.

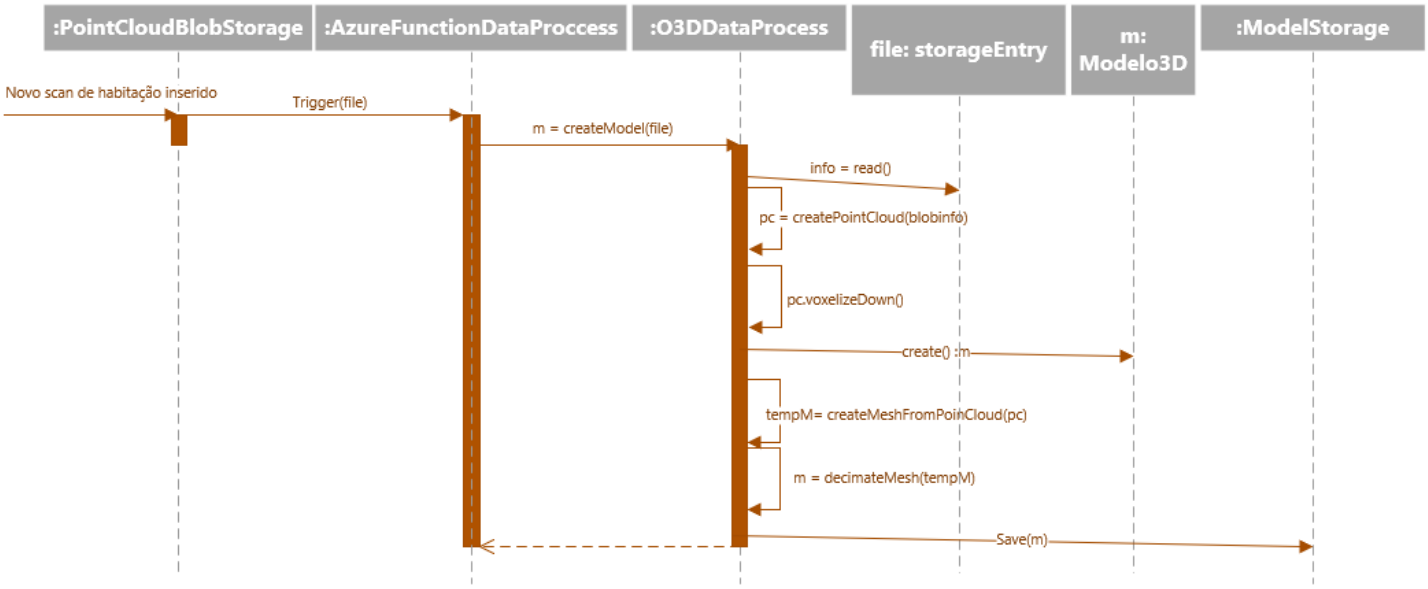


Figura 5-3 – Diagrama de sequência: processar informação dos imóveis em modelos 3D

##### 5.2.1.2.2 Visualizar imóveis

O processo de visualização de imóveis exige interação e criação de uma interface com o utilizador. O principal objetivo deste processo é o de fornecer ao utilizador uma visualização do imóvel escolhido pelo cliente. Para isso, o processo desenrola-se da seguinte forma:

- O cliente inicia o processo de visualizar o imóvel 3D;
- O browser requere a escolha do imóvel a visualizar ao utilizador;

- O utilizador escolhe o imóvel que pretende visualizar;
- O browser obtém a informação do modelo 3D do imóvel existente na *storage*;
- O browser cria a cena virtual e recria nela o modelo 3D do imóvel;
- O browser mostra a cena ao utilizador.

Na Figura 5-4 é possível obter uma perspetiva do processo de uma forma visual.

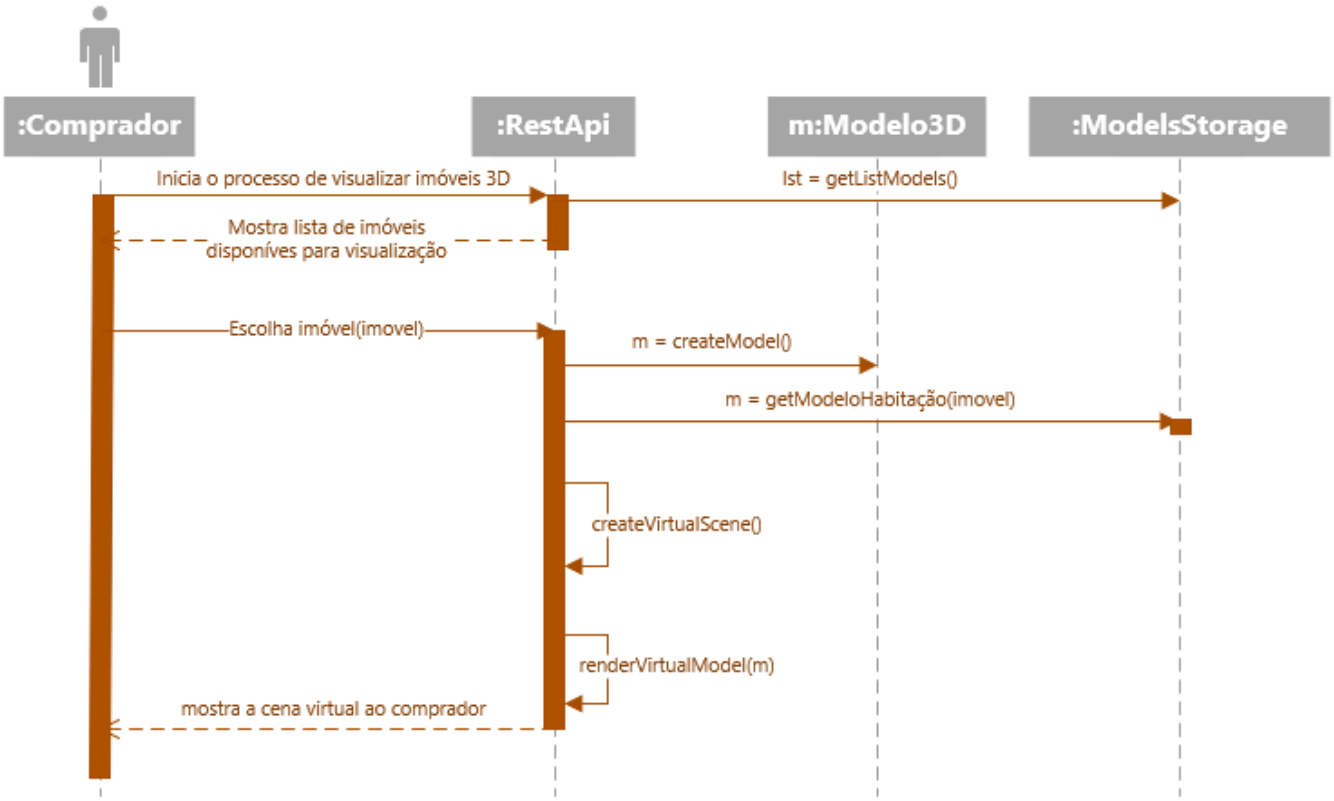


Figura 5-4 – Diagrama de sequência: visualização de imóveis

5.2.1.2.3 Scan da habitação

O processo da habitação tem uma vertente extra sistema que necessita de uma scanner para realizar a captura de informação necessária para os restantes processos. Após a obtenção do scan, o utilizador terá de inserir esse scan num browser que envia a informação para uma *storage*, o que despoleta o processo de processar informação dos imóveis.

- O cliente inicia o processo de scan da habitação;
- O módulo de Scan efetua a digitalização do imóvel;
- O Scanner retorna os dados relativos à digitalização do imóvel;

- O utilizador insere os dados obtidos no sistema;

- Os dados obtidos são guardados na base de dados e retorna sucesso da operação

A Figura 5-5 representa visualmente o processo descrito anteriormente.

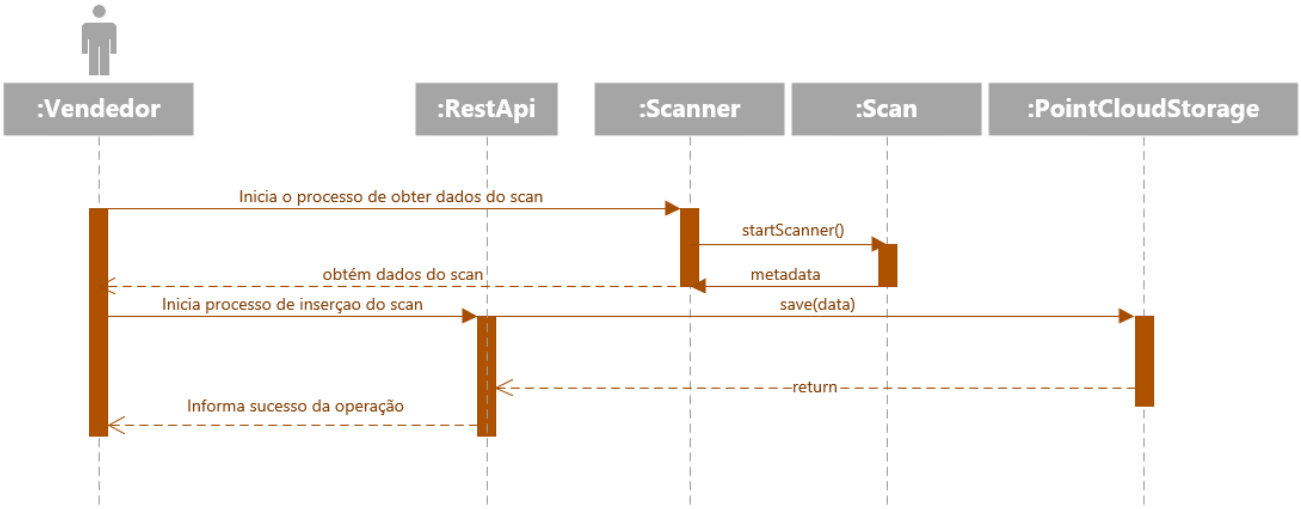


Figura 5-5 – Diagrama de sequência de scan da habitação

#### 5.2.1.3 Vista de implantação

Para a implantação da solução, a web app de interação com o utilizador (*Website* para visualização e para recolha de informação) estará numa *webapp*, e os dados para persistências estarão em *storages.* Quanto à componente de processamento/ transformação dos dados de *point clouds* para modelo 3D esta será colocada num container de Docker

Deste modo, a implantação está descrita visualmente na Figura 5-6, com os nós:

- *Web App* – nó que terá como objetivo a implantação da webapp de scan e visualização dos dados;

- *Function* – nó que irá chamar o Docker container de processamento dos dados em Open3D. Esta função será despoletada através da introdução de dados na *blob*;

- *Storage* – nó que terá duas *blob storages*, como forma de persistência dos dados da solução.

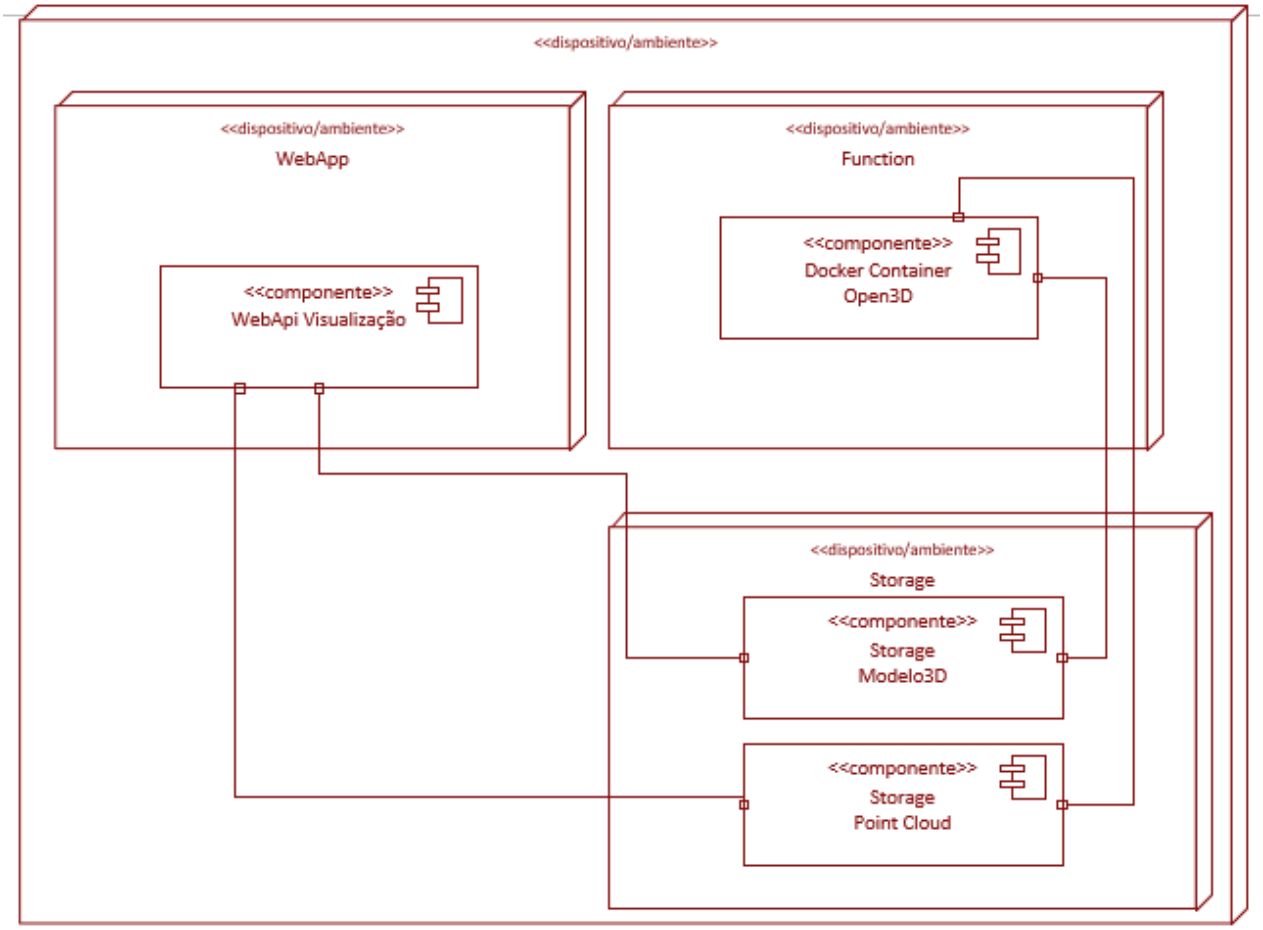


Figura 5-6 – Diagrama de implantação

# 6 Construção da solução

O presente capítulo tem como objetivo apresentar a construção a solução desenvolvida.

## 6.1 Detalhes da prova de conceito desenvolvida

Tendo por objetivo um produto complexo e que requeria pesquisa e procura de informação acerca do tema, foi decidido realizar uma prova de conceito que se focasse na reconstrução e recriação da nuvem de pontos. Esta recriação deveria resultar num modelo 3D eficiente e com uma qualidade aceitável.

Como referido no capítulo Abordagens alternativas, a tecnologia a utilizar na leitura de *point clouds* e criação da malha poligonal será o Open3D.

A possibilidade de utilizar métodos *out of the box* e a possível interação com as restantes bibliotecas já existentes de *Python*, permitem uma versatilidade e adaptabilidade para o problema aqui enfrentado.

Para efetuar melhorias à malha poligonal gerada foi utilizado o *Simplygon,* uma biblioteca que permite a redução do modelo 3D. Esta redução é efetuada através da redução dos polígonos existentes da malha, tendo atenção às caraterísticas geométricas e cor existente.

O *Simplygon* permite ainda mapear texturas e cores existentes na malha original e transcrevê-las para o modelo 3D reduzido.

### 6.1.1 Objetivos da prova de conceito

Tal como referido anteriormente (no capítulo Objetivo) esta prova de conceito tem como finalidade analisar a possibilidade de recriação de espaços interiores, através de scans, que culminam na criação de um modelo 3D.

Deste modo, os principais aspetos que a prova de conceito terá de se focar são:

- Pré-processamento da nuvem de pontos recolhida;
- Criação do modelo 3D através da nuvem de pontos;
- Melhorar o modelo 3D;
- Utilização do modelo numa cena renderizada em browser.

Esta prova deve também dar resposta às hipóteses colocadas anteriormente no capítulo Objetivo, referente à existência de possibilidade de recriação da *point cloud* e da forma como a realizar.

## 6.2 Prova de conceito

Tendo por foco a criação e melhoria do modelo 3D, este processo é dividido em 3 fases: pré-processamento da *point cloud*, criação do modelo 3D primordial a partir da *point cloud*, melhorias ao modelo 3D.

### 6.2.1 Pré-processamento da point cloud

Tal como referido anteriormente, o processo de leitura e pré-processamento da *point cloud* é efetuado utilizando as funcionalidades disponibilizadas pelo *Open3D*.

Para efetuar o processo de leitura da *point cloud* é necessário que esta se encontre nos formatos xyz, xyzrgb, pts, ply ou pcd. Utilizando funções do *read_point_cloud* e *write_point_cloud*, o *Open3D*, permite importar/exportar as *point clouds*. A informação relativa às *point cloud* ficam armazenadas em *arrays numpy*.

Ao efetuar a importação da *point cloud* é ainda encontrado o centro da mesma, como forma de permitir centralizá-la na cena. Para realizar a centralização dos pontos é efetuada a translação com um vetor oposto ao vetor formado entre a origem do referencial (0,0,0) e o centro da *point cloud.*

Após a importação, os pontos da *point cloud* são pré-processados, sendo feita a redução da *point cloud*, remoção de possíveis outliers que não façam sentido à cena e estimadas as normais dos pontos (caso estas normais não estejam já presentes na *point cloud*).

#### 6.2.1.1 Redução da *point cloud*

Para efetuar a redução da *point cloud* é utilizado o método de *down sample* a partir de *voxels*, onde é definido o tamanho desejado para o do lado do voxel. Assim sendo é realizada a redução do número de pontos através do *clustering* de pontos vizinhos de forma que o tamanho dos pontos/*voxels* seja o desejado. Para a realização deste *clustering*, as características dos pontos, tais como cores e normais são interpoladas entre os pontos do *cluster* que geram cada *voxel*.

#### 6.2.1.2 Remoção de outliers

O processo de remoção dos *outliers* existentes na nuvem é realizado com a utilização do método *remove_statistical_outlier* existente na biblioteca *open3d.* Este processo identifica os *outliers* através do número de pontos vizinhos ao ponto *outlier* e do desvio padrão das distâncias entre eles. Caso o desvio padrão seja superior ao rácio permitido, o ponto é identificado como *outlier* e é removido da *point cloud*.

#### 6.2.1.3 Estimativa das normais dos pontos

A existência de *point clouds* que não apresentam informação sobre os vetores normais dos pontos, o que implica que estes vetores sejam estimados. São utilizados algoritmos que realizem uma estimativa dos mesmos vetores, como forma de poderem ser utilizadas na recriação em malha poligonal, numa fase posterior do projeto.

A estimativa das normais dos pontos é efetuada através de um algoritmo que calcula a covariância das normais dos pontos em relação à sua vizinhança. Este algoritmo produz então dois tipos de normais em direções opostas uma da outra, sendo que a escolhida depende da estrutura global da nuvem.

Todo o processo referido anteriormente foi implementado recorrendo à biblioteca *Open3d.* Esta implementação está documentada sobre forma de diagrama de sequência na Figura 6-1.

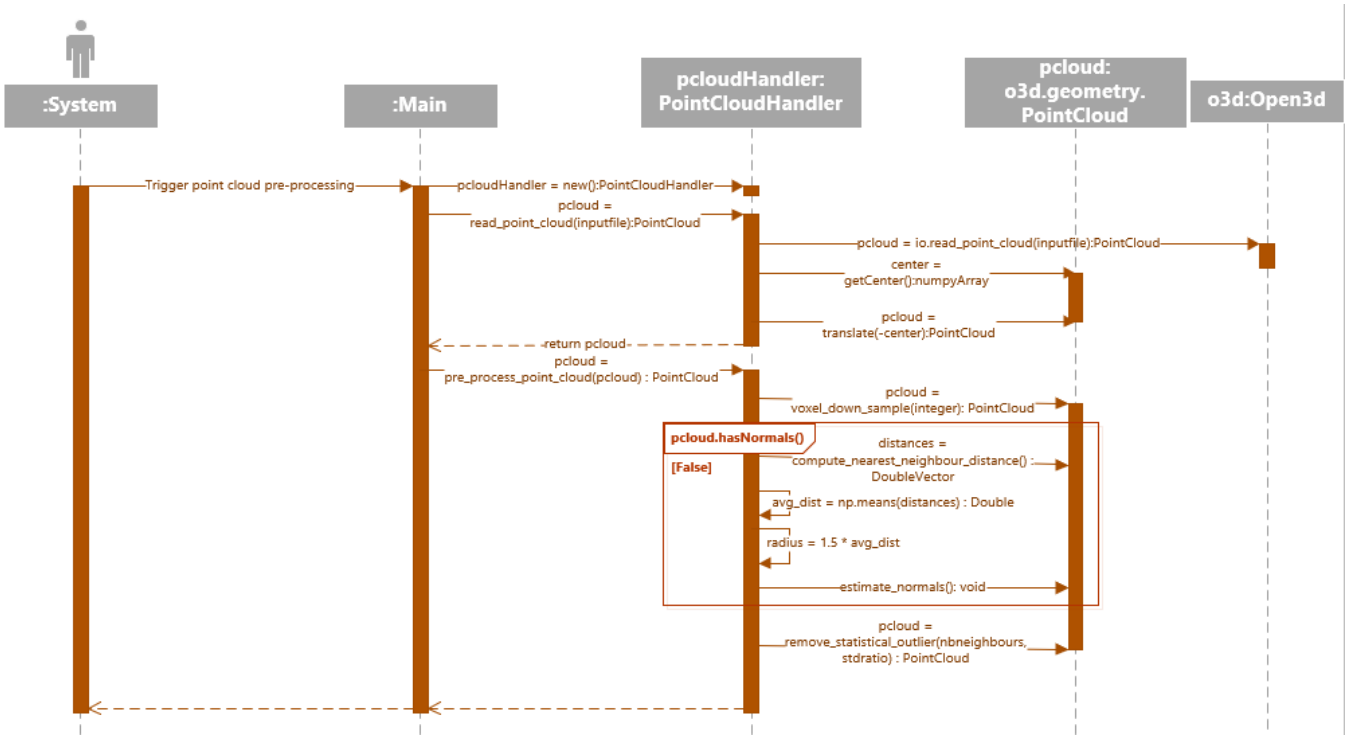


Figura 6-1 – Diagrama sequência: pré-processamento da *point cloud*

Um exemplo de uma *point* cloud resultante destas transformações é apresentada na Figura 6-2, onde é possível verificar a existência de um milhão de pontos e nenhuma face.

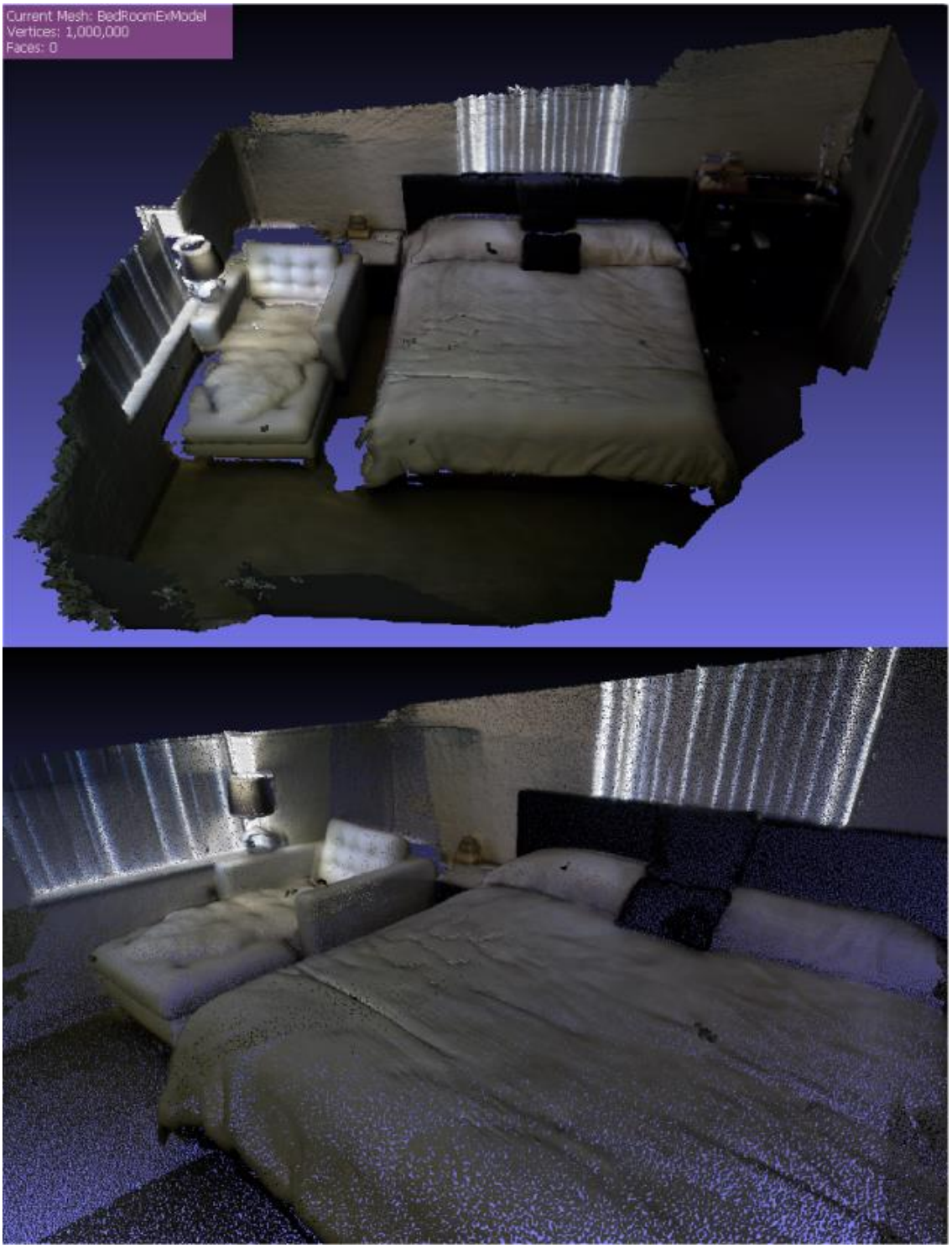


Figura 6-2 – *Point cloud* exemplo referente a um quarto

### 6.2.2 Criação do modelo 3D

Após efetuado o pré-processamento da *point cloud,* esta está pronta para ser utilizada na reconstrução das superfícies, gerando assim um modelo 3D a partir de malha poligonal.

Esta criação do modelo pode ser efetuada de diversas formas, sendo que as abordagens utilizadas nesta prova de conceito foram o *Ball Pivoting Algorithm (BPA)* e a Reconstrução de *Poisson*.

Foram testadas ambas as reconstruções. A reconstrução pelo algoritmo de *Ball Pivoting* foi efetuada com um raio mínimo para a bola é equiparável à distância média entre os pontos e o máximo a duas vezes essa distância, onde a reconstrução resultou numa malha poligonal com 18363 vértices e 30299 faces (como é apresentado na Figura 6-3).


Figura 6-3 – Exemplo de reconstrução por *Ball Pivoting Algorithm*

A outra opção para a reconstrução da *point cloud* é o algoritmo de *Poisson*. Esta reconstrução foi efetuada com depth de 15 e com auxílio a *linear fit*. O resultado obtido por esta reconstrução é apresentado na Figura 6-4.

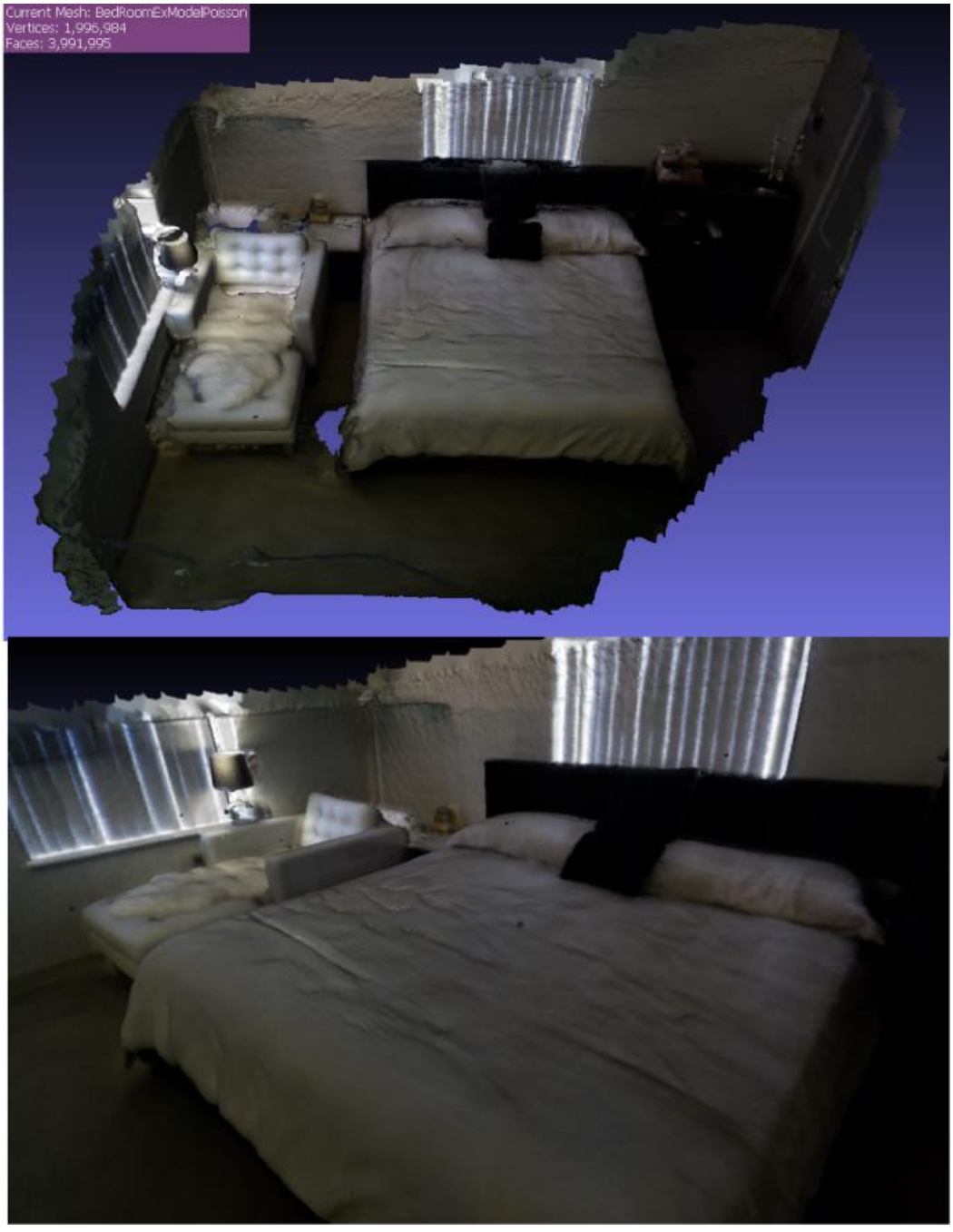


Figura 6-4 - Exemplo de reconstrução de *Poisson*

Após análise dos resultados obtidos, foi escolhida a reconstrução de *Poisson* como o método principal a utilizar na solução.

Após a utilização destes métodos é gerada uma malha poligonal em que o vértice detém a cor. Existe a possibilidade de na geração da malha possam existir triângulos ou vértices que se

encontrem em duplicado ou que existam arestas que estejam em duplicado em diversos triângulos e que necessitam de ser removidos.

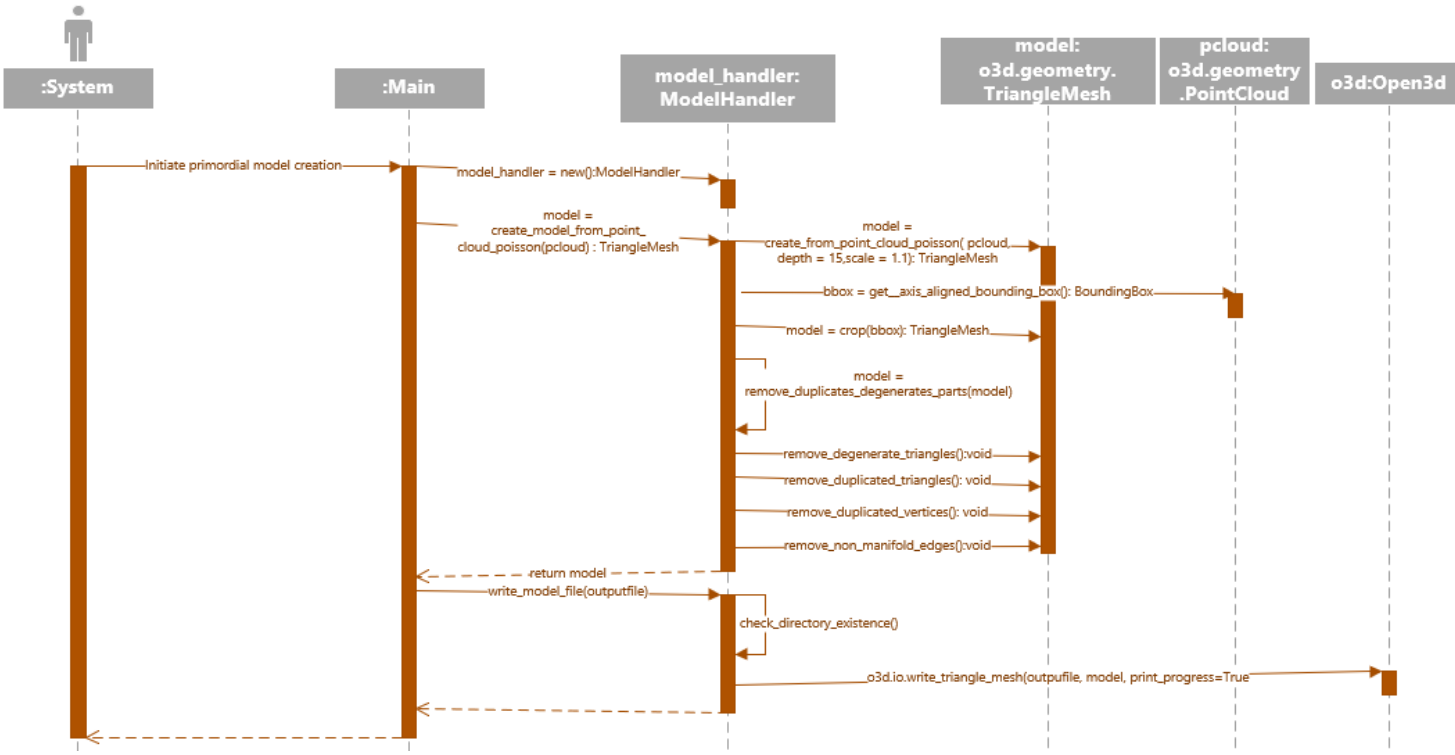


Figura 6-5 – Diagrama de sequência: Criação modelo 3D.

Após a criação dos modelos foi encontrada a necessidade de reduzir o impacto que os modelos têm, de forma a diminuir o espaço que estes ocupam em disco e a aumentar a performance que estes apresentam quando renderizado no browser. Com vista nesta finalidade foi necessário implementar um passo que consistia em melhorar o modelo 3D.

### 6.2.3 Melhorias ao modelo 3D

Tendo sido identificada a necessidade de melhorar o modelo 3D, foi optado a utilização do Simplygon, uma ferramenta que permite melhorar e otimizar a performance e recursos utilizados pelo modelo 3D.

A forma de melhorar o modelo 3D visa na redução dos polígonos (triângulo, neste caso). Para efetuar esta redução, o *Simplygon* disponibiliza na sua API métodos para efetuar esta redução (o *ReductionProcessor*) que permite a definição dos objetivos para a redução. Estas definições são passadas para o objeto da classe *SimplygonReductionSettings*, que permite a definição de parâmetros como: o rácio de triângulos, o número de triângulos do modelo final, o desvio máximo que o resultado final pode apresentar em relação ao modelo inicial, entre outras.

Deste modo foram realizadas reduções nos modelos, onde estas reduções tinham como objetivo diminuir os triângulos do modelo 3D até um mínimo de 100 000 triângulos ou até

este apresentar cerca de 10% dos triângulos que a reconstrução inicial, feita pelo open3d, gerou (Figura 6-6).

```
# Set reduction target to triangle ratio with a ratio of 10%.
#void SetReductionTargets( Simplygon::EStopCondition stopCondition , bool useTriangleRatio , bool useTrian
sgReductionSettings.SetReductionTargets( Simplygon.EStopCondition_Any, True, False, False, False )
sgReductionSettings.SetReductionTargetTriangleRatio( 0.1 )
sgReductionSettings.SetReductionTargetTriangleCount(100000)
```

Figura 6-6 – Código que define as condições de redução do modelo 3D

O processo de melhorar o modelo 3D por via da simplificação do mesmo é demonstrado pelo diagrama da Figura 6-7.

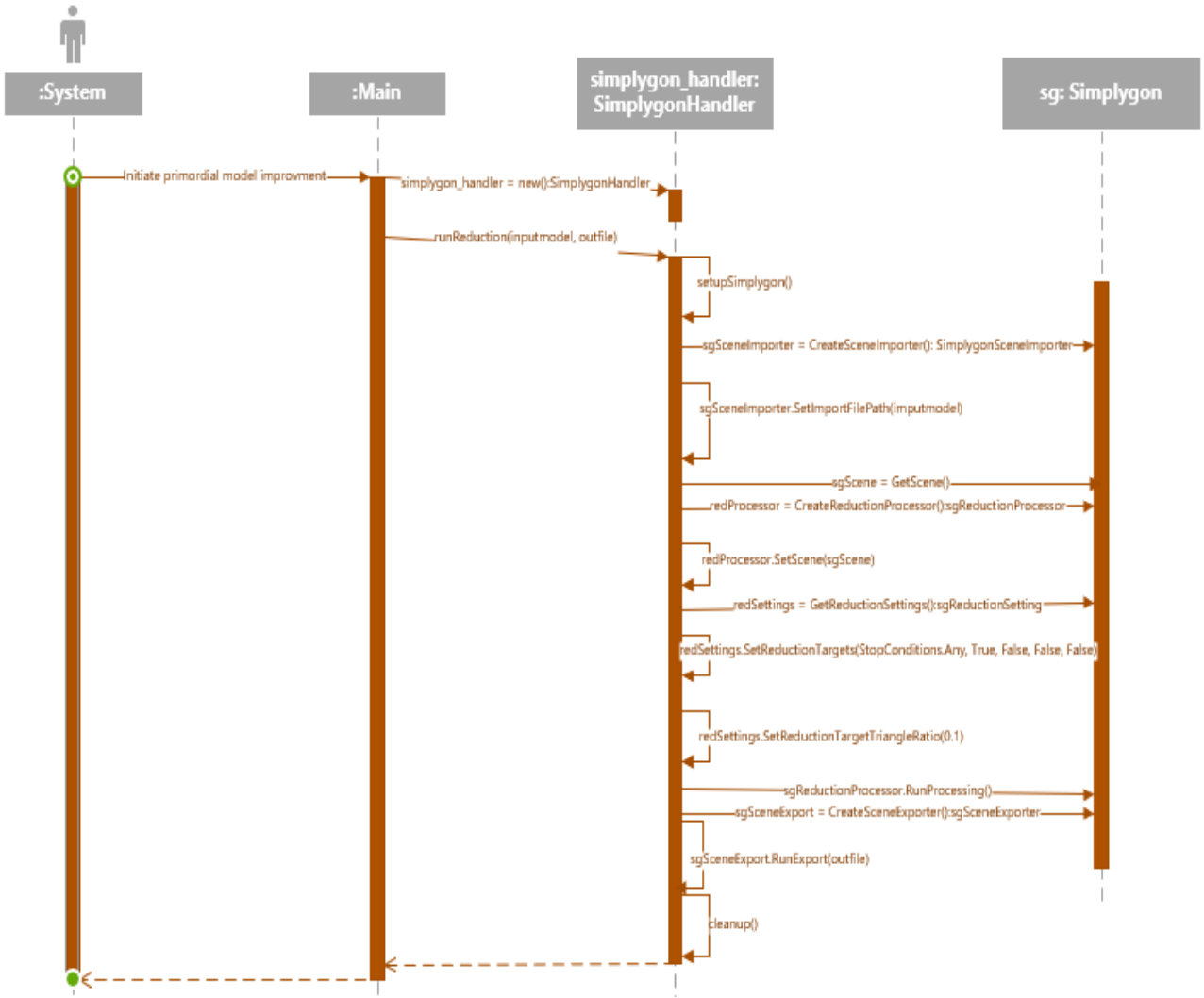


Figura 6-7 – Diagrama sequência: Simplificação do modelo 3D.

### 6.2.4 Visualização dos modelos 3D num browser

Após a criação dos modelos 3D, a sua visualização pode ser efetuada em software como o *Meshlab* ou *Blender* (que permitem importar/editar modelos 3D através de interface gráfica) ou, como foi utilizado nesta prova de conceito, através da renderização num browser, tirando partido das potencialidades do ThreeJs.

Para renderizar estes modelos no *Threejs*, foi necessário implementar uma página *HTML* que contém um *Javascript* utilizando o *Threejs* como biblioteca para renderizar o modelo, através do *WebGL*.

Esta implementação tirou partido de bibliotecas já existentes no *Threejs* para criação de uma cena. Numa primeira instância é criada uma função *init()* onde são criados todos os aspetos e elementos importantes para a cena (câmara, renderer, luminosidade e controlos).

A camara tira partido da *PerspectiveCamera* disponibilizada pela ferramenta, criando assim uma camara *default* para visualizar o que ocorre na cena, que utiliza uma projeção de perspetiva que simula o olho humano.

O *renderer*, tal como referido anteriormente o *Threejs,* utiliza o *WebGl*, logo este é gerado a partir da chamada à função *WebGLRenderer().*

Ao nível de controlos na cena, e com o objetivo de dar liberdade de movimento para analisar o modelo 3D a importar, foi optado pela utilização dos *FirstPersonControls()* do *ThreeJS*.

Para testar o modelo3D e ser possível visualizá-lo, uma vez que estes modelos estão no formato de *GLTransmissionFormat* (.glb), estes foram importados a partir da biblioteca GLTFLoader. Uma vez que os objetos não apresentam texturas, sendo que a cor está presente nos vértices do modelo, é necessário alterar o material por defeito do método (que iria utilizar as texturas presentes no ficheiro) e indicar que este utilizará as cores dos vértices. Para isso foi gerado um *MeshBasicMaterial* em que o parâmetro *vertexColors* está definido e é indicado como a cor a utilizar para renderizar o modelo.

Existe ainda uma biblioteca (*Stats.*js) que permite mostrar as estatísticas da performance da cena, o que é útil para avaliar qual o impacto que o modelo 3D cria na visualização da cena.

Após estas configurações, a cena está apta para ser colocada no browser, sendo assim possível analisar o desempenho e a qualidade do modelo 3D neste ambiente.

Na Figura 6-8 é apresentado o resultado final da cena, onde é possível visualizar um modelo 3D de exemplo.

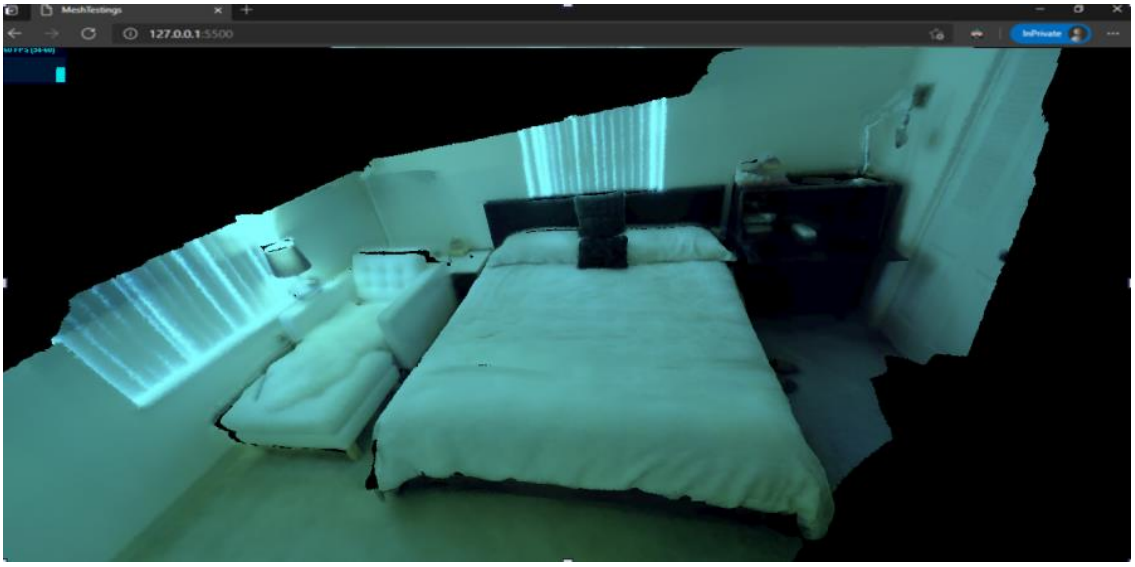


Figura 6-8 – Exemplo visualização do modelo 3D num browser

## 6.3 Testes

Os testes, de uma forma generalizada para engenharia de software, servem para confirmar que o código implementado executa e tem os resultados esperados e é um método de deteção de anomalias no programa. [55]

Existem diferentes tipos de testes que podem ser efetuados, sendo que neste caso foram realizados:

- Testes unitários;
- Testes aceitação;

### 6.3.1 Testes unitários

Um teste unitário é um teste que se foca numa componente do programa, tal como métodos e objetos implementados. Estes testes devem ser executados sobre os métodos individuais que foram implementados, de forma a verificar que se comportam da forma expectável. [55]

Posto isto, foram realizados testes unitários às classes que manipulam as *point cloud* e *os modelos*, nomeadamente o *PointCloudHandler* e o *ModelHandler.*

Estes testes foram criados através da *framework pytest* que é uma ferramenta de criação e testes em *Python*.

Na Figura 6-9 é apresentado um exemplo de um teste unitário efetuado à classe *PointCloudHandler*. Os testes apresentados tinham por função garantir que o pré-

processamento da *point cloud* era efetuado com sucesso, ou seja, que os vetores normais tinham sido gerados e que o número de pontos tinha sido diminuído.

```
def test_pre_process_point_cloud_normals_created(self, dummy_point_cloud_handler, dummy_point_cloud):
    pytest.point_cloud_handler.set_cloud(pytest.point_cloud)
    result_cloud = pytest.point_cloud_handler.pre_process_point_cloud()
    assert result_cloud.has_normals() == True
    assert len(result_cloud.points) <= len(pytest.point_cloud.points)
```

Figura 6-9 - Exemplo de teste unitário ao *PointCloudHandler*

Após a criação dos testes, estes são executados através da linha de comandos executando a instrução '*pytest*', sendo que o resultado se encontra na Figura 6-10.

```
(myenv) C:\Users\andre.lima\Desktop\Estagio2020\GitEstagio\estagio\Project>pytest --disable-pytest-warnings
============================================= test session starts =============================================
platform win32 -- Python 3.7.9, pytest-6.2.4, py-1.10.0, pluggy-0.13.1
rootdir: C:\Users\andre.lima\Desktop\Estagio2020\GitEstagio\estagio\Project
collected 9 items

tests\test_model_handler.py ....                                                                      [ 44%]
tests\test_point_cloud_handler.py .....                                                               [100%]

==================================== 9 passed, 1 warning in 123.00s (0:02:03) =====================================
```

Figura 6-10 – Resultado execução testes unitários

Analisando o resultado anterior, uma vez que os testes unitários foram executados com sucesso, é possível concluir que o código implementado se comporta da forma esperada.

### 6.3.2 Testes de aceitação

Os testes de aceitação representam o interesse do cliente, dando-lhe assim garantias que as funcionalidades e objetivos do projeto foram realizadas e estão de encontro às expectativas do cliente. [56]

Tendo em conta que os testes de aceitação vão de encontro ao que o cliente desejava, ou seja, aos requisitos definidos, a Tabela 6-1 foi elaborado como artefacto dos testes de aceitação.

Tabela 6-1 – Testes de aceitação

| Requisito | Resultado |
|---|---|
| Pré-processamento da *point cloud* | Cumprido |
| Criação do modelo 3D | Cumprido |
| Melhorias ao modelo 3D | Cumprido |
| Recriação do modelo num browser | Cumprido |

# 7 Experimentação e Avaliação

Este capítulo tem por foco a conceção do processo de experimentação e avaliação da solução.

## 7.1 Objetivo

O objetivo da experimentação e avaliação é o de avaliar o resultado do trabalho desenvolvido durante o projeto.

A avaliação é importante para analisar os programas e sistemas de forma a torná-los mais eficientes e com uma performance mais rápida. A otimização da performance é importante para obter uma aplicação funcional e eficiente. Para ser possível esta otimização, é importante monitorizar, analisar e avaliar a performance e identificar formas de a melhorar. [57]

A alteração do sistema é realizada seguindo os seguintes aspetos:

- Parâmetros de criação dos modelos 3D das habitações, criando alternativas na criação dos modelos e escolha de qual utilizar;
- Indicadores de performance de execução da cena 3D no browser, e como melhorar a essa performance e eficiência da cena;

## 7.2 Indicadores e fontes de informação

Como forma de avaliar a solução obtida é necessário analisar indicadores que indiciem os níveis de qualidade e eficiência. Para isso são utilizados indicadores de – desempenho, monitorização, cumprimento dos objetivos propostos.

Para avaliar esta solução, são utilizados:

- Simplificação dos modelos 3D e impacto no tamanho/qualidade;
- Tempo execução dos processos implementados;
- Tempo de renderização da cena no browser;
- Tamanho em disco necessário para armazenar o modelo;
- Qualidade percecionada dos modelos 3D gerados.

A informação referente à cena renderizada no browser é obtida diretamente através de um monitor de performance existente para *Threejs*, o *stats.js* [58].

Quanto ao tamanho em disco necessário para armazenar o modelo é feita uma comparação com o tamanho da *point cloud inicial* e o modelo 3D gerado, bem como o efeito que a redução deste modelo teve neste aspeto.

Para medir os tempos de execução das etapas de criação e simplificação dos modelos, foi utilizado o *Jupyter Notebooks* com apoio da biblioteca *autotime* [59].

Os processos de recriação da *point cloud* foram executados em duas máquinas distintas:

- PC1 – CPU: Intel i5-6200U 2.30GHz (4CPUs), RAM: 16GB, GPU: Intel HD Graphics 520, SO: Windows;
- PC2 – CPU: Intel i7-6700 3.4GHz (8CPUs), RAM: 32GB, GPU: Nvidia GTX1080 12GB, SO: Linux.

Em ambos os computadores foi realizada a execução do código para pré processamento e criação de modelos 3D. Uma vez que o simplygon não tem suporte atual para arquiteturas Linux, o processo de simplificação do modelo 3D foi executado somente no PC1 referido anteriormente, sendo importante verificar a redução efetuada ao modelo (no que toca a atributos do modelo 3D e ao seu tamanho de armazenamento). A execução e testes dos modelos simplificados no browser foram também efetuados no PC1.

## 7.3 Metodologia de avaliação

Na metodologia esteve envolvida a pessoa que desenvolveu o projeto e foram seguidos os passos:

- Reprodução simplificada do sistema inicial;
- Desenvolvimento do sistema referente à solução pretendida com várias opções de criação dos modelos 3D;

- Verificado para a solução implementada o resultado das grandezas, averiguando a duração;
- Tempo e impacto que cada modelo teve na recriação da cena virtual;
- Análise dos resultados obtidos.

## 7.4 Experiências efetuadas

Foram realizadas experiências e analisados os resultados em três fases da prova de conceito: modelos 3D inicialmente gerados; simplificação dos modelos; renderização dos modelos no browser.

### 7.4.1 Análise geração modelos 3D iniciais

Para realizar a geração dos modelos 3D é necessário realizar um pré-processamento das *point clouds*. Para estes testes foi utilizada a *point cloud* de um quarto, com 1 000 000 pontos. Este número de pontos foi alterado nas execuções através do processo de *downsampling*, permitindo assim verificar o impacto da diferença de tamanhos da *point cloud* no pré-processamento da mesma.

O processo mais demorado de pré-processamento é a estimativa de normais que aumenta consoante o tamanho da *point cloud*, *clouds* maiores demoraram mais tempo.

Nas experiências efetuadas no PC1, o tempo máximo obtido para o pré-processamento foi de cerca 12 segundos para uma nuvem de 1 000 000 de pontos, e de 5 segundos para uma nuvem de 20 000 pontos. Quanto aos tempos obtidos no PC2, com melhores recursos computacionais, foi obtido uma redução dos tempos de pré-processamento em cerca de 40%, sendo que para os 1 000 0000 de pontos foram necessários apenas 8 segundos e para a *point cloud* de menores dimensões apenas 2 segundos.

Os tempos obtidos nas experiências estão disponíveis em anexo, na secção de experiências e avaliação.

Para analisar a geração dos modelos 3D foi tido em conta a qualidade percecionada dos modelos consoante os diversos parâmetros. Para o caso, as opções seriam a utilização do *Ball Pivoting Algorithm* vs. *Reconstrução de Poisson* (Figura 7-1).

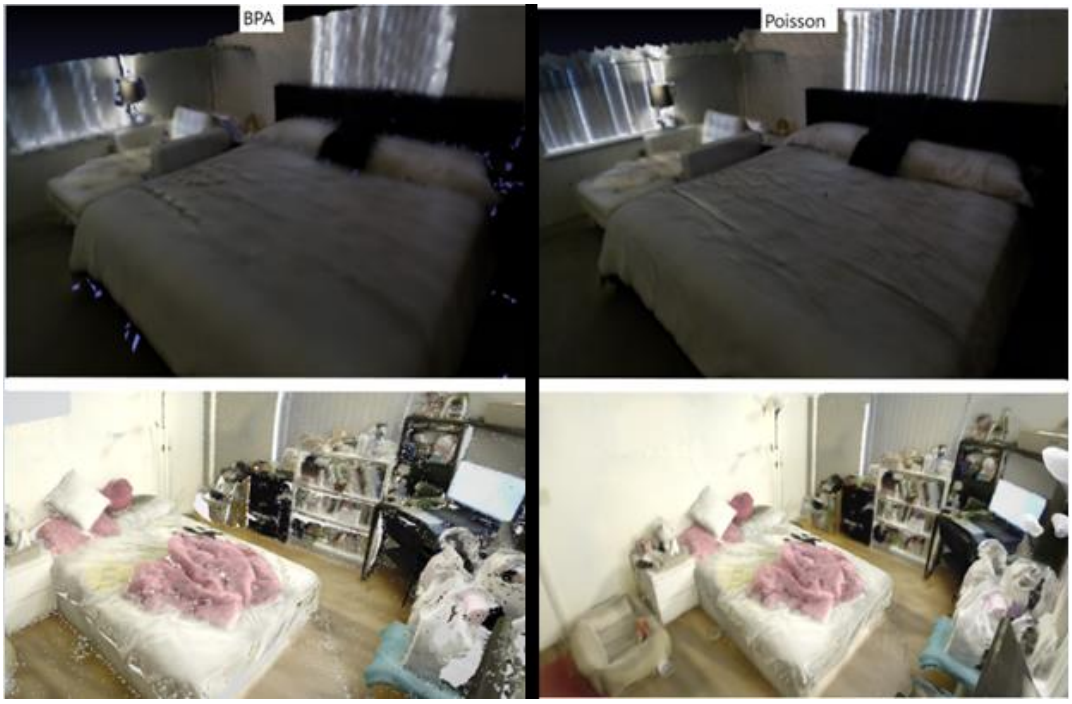


Figura 7-1 – Comparação de reconstruções: *Ball Pivoting Algorithm vs. Poisson*.

Foi então concluído que a reconstrução de *Poisson* seria a mais indicada, uma vez que permitia reconstruir a *point cloud* inicial com menos falhas/buracos na malha poligonal, quando comparada com o algoritmo de *Ball Pivoting*.

Uma vez definida que a reconstrução de *Poisson* é a que seria utilizada, é necessário aprimorar os parâmetros da reconstrução. O parâmetro mais focado foi o da profundidade (*Depth)* a que o método iria reconstruir. Quanto maior o parâmetro de profundidade, maior o detalhe e a precisão do modelo reconstruído. Estas diferenças na qualidade estão presentes nas reconstruções apresentadas na Figura 7-2.


Figura 7-2 – Comparação do parâmetro *depth* da reconstrução de *Poisson*.

Assim sendo, foi testada a performance da execução em ambos os PC's referidos anteriormente, e foi escolhido utilizar uma profundidade com o valor 15, cuja utilização de memória necessária era próxima dos 9GB para modelos mais pequenos e cerca de 13GB para modelos de maior proporção.

Quanto aos tempos de execução para a geração dos modelos 3D, ao utilizar a reconstrução de *Poisson*, é possível verificar a diferença em executar o mesmo código nos dois ambientes disponíveis. Para o caso do PC1, para reconstruir um modelo a partir de 1 000 000+- são necessário no máximo 6 minutos e 23 segundos apenas para a execução da reconstrução. Para um modelo reconstruído a partir de 20 000+-, são necessários 1 minutos e 07 segundos.

Por outro lado, no PC2, a reconstrução foi mais rápida, sendo que para serem reconstruídos 1 000 000+- são necessários 2 minutos e 11 segundos. Por outro lado, reconstruir uma *point cloud* de menores dimensões (20 000+-) apenas necessita de 45 segundos.

Desta forma é possível evidenciar a importância dos recursos computacionais na celeridade dos processos de pré-processamento e reconstrução dos espaços.

Este processo de criação dos modelos varia no seu tempo de execução consoante o tamanho original da *point cloud*, porém é possível evidenciar que para 1 000 000 de pontos, este processo necessita de no mínimo 6 minutos para o PC1 e 2 minutos no caso do PC2.

### 7.4.2 Simplificação dos modelos 3D

O objetivo da simplificação dos modelos era torná-los mais leves a nível computacional e a nível de armazenamento em disco.

Para tornar mais leve a nível de computação é necessário diminuir o número de vértices/triângulos que estão presentes nos modelos 3D. Isto terá um impacto também no tamanho do ficheiro que armazena estes mesmos modelos, uma vez que a informação que será guardada é, também ela, reduzida.

Numa primeira fase, foram analisados os parâmetros de redução disponibilizados pelo *Simplygon*. Destes parâmetros foi escolhido a percentagem de triângulos a manter (*TriangleRatio)* do modelo 3D inicial, sendo que foram efetuados testes com diferentes valores (nomeadamente, 70%, 50% e 10%). Um exemplo das reduções obtidas está presente na Figura 7-3.

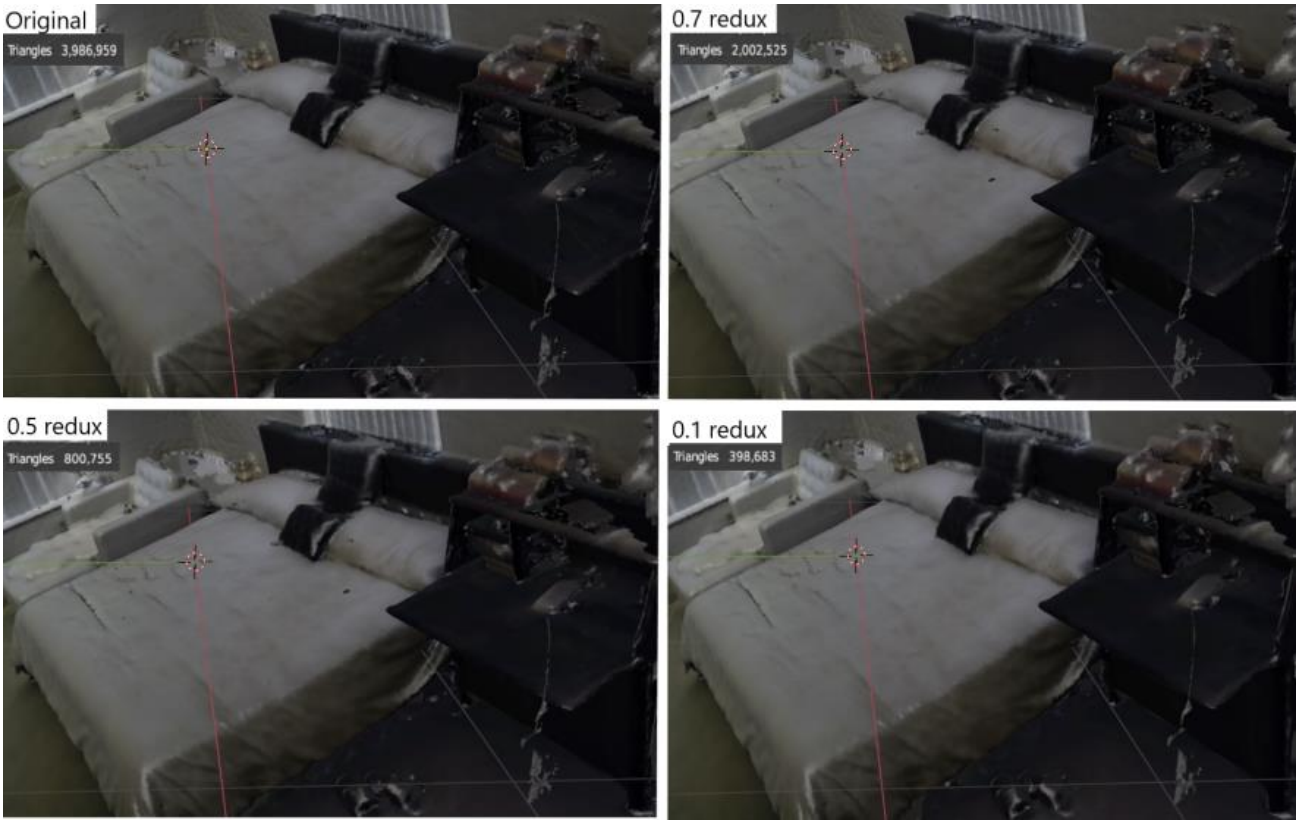


Figura 7-3 – Exemplo do mesmo modelo 3D reduzido com diferentes *TriangleRatio*.

Ao analisar os resultados das diferentes reduções, foi chegada à conclusão de que a definição não era assim tão afetada a nível de qualidade pelas reduções efetuadas no *Simplygon*, pelo que foi definido utilizar o valor de 10% para as reduções posteriores.

Deste modo são comparados alguns modelos 3D em dois pontos da prova de conceito: quando estes são recriados a partir da *point cloud* e após a execução do processo de simplificação. Os atributos de cada modelo são apresentados na Tabela 7-1.

Tabela 7-1 – Dados dos modelos 3D antes e após a simplificação.

| Modelo | Vert_i | Tri_i | Mem_i (MB) | Vert_f | Tri_f | Mem_f (MB) |
|---|---|---|---|---|---|---|
| **Bedroom0** | 1.99M | 3.98M | 152.2 | 233 248 | 398 683 | 13.4 |
| **Bedroom1** | 3.70M | 7.93M | 302 | 445 559 | 793 483 | 26.0 |
| **Bedroom2** | 2.62M | 5.23M | 196 | 327 129 | 523 252 | 18.5 |
| **OfficeHall** | 2.59M | 5.17M | 197 | 294 946 | 517 574 | 17.1 |
| **LivingRoom** | 2.58M | 5.16M | 199 | 281 863 | 515 827 | 16.6 |

Vert_i: nº vértices antes da recriação, Tri_i: nº triângulos antes da recriação, Mem_i: espaço no disco pelo modelo ocupado antes da simplificação, Vert_f: nº vértices após simplificação, Tri_f: nº triângulos após simplificação, Mem_f: espaço no disco ocupado pelo modelo após a simplificação.

Os modelos 3D simplificados referenciados na tabela anterior foram obtidos através da redução dos modelos que resultaram da recriação inicial, com um rácio de 10% dos triângulos.

Na Figura 7-4 é possível ver o resultado da simplificação da *Bedromm0* (presente na tabela anterior). Nesta figura e possível verificar que não existe uma diminuição na qualidade percecionada do modelo 3D, porém ao analisar a malha que constitui estes modelos, é possível verificar a simplificação desta malha, principalmente nos planos de maior área, onde ocorreu a aglomeração/substituição de triângulos de menores dimensões por triângulos coplanares de maior dimensão. Este processo permite diminuir o peso computacional que a malha do modelo 3D simplificado apresenta quando comparada com a original.

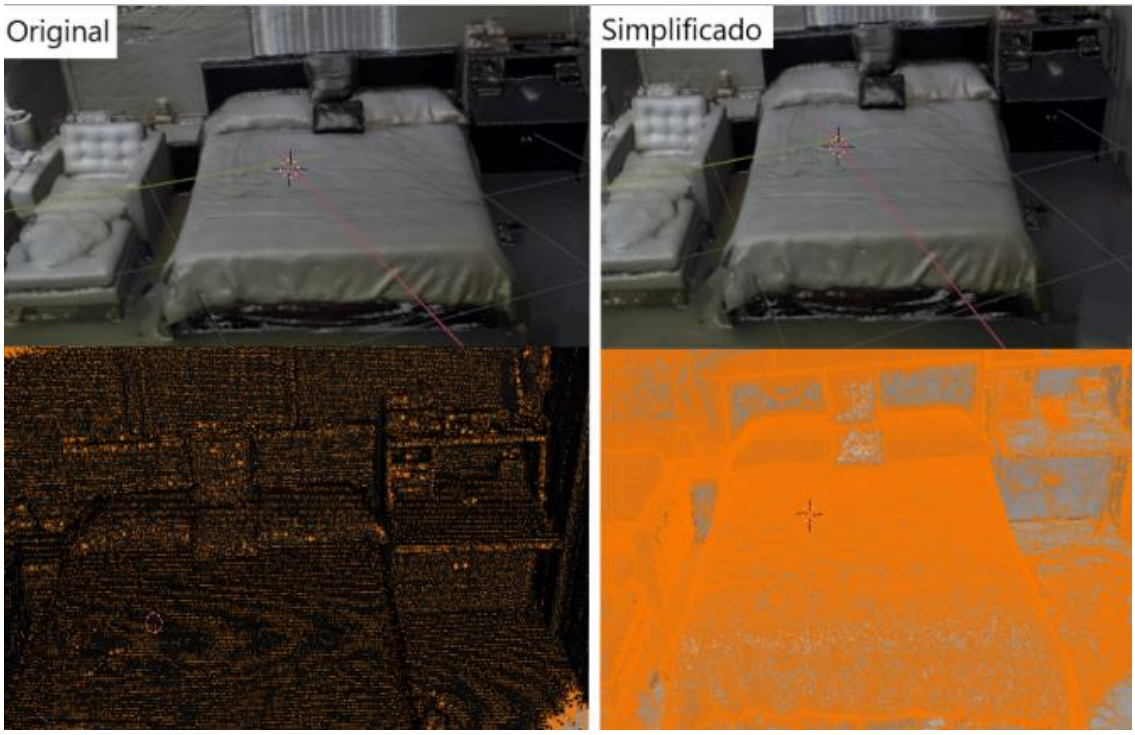


Figura 7-4 – Exemplo simplificação da *BedRoom0*.

Na imagem anterior é possível evidenciar à esquerda, o modelo 3D original, onde é complicado identificar triângulos na malha poligonal. Na direita, o modelo 3D simplificado, onde já é possível identificar alguns triângulos na malha poligonal constituinte.

O processo de simplificação a partir do Simplygon está apenas disponível para sistemas operativos Windows, deste modo, foram apenas efetuados testes no PC1. Quanto aos tempos obtidos pelo Simplygon, o processo demora no mínimo 40 minutos a executar para modelos com cerca de 1 500 000 de pontos e cerca de 3 000 000 de triângulos.

Este processo de simplificação pode também ser efetuado a partir do Open3D, de uma forma mais simplificada (aumenta apenas o tamanho dos triângulos, sem ter em conta o detalhe de

certas regiões), mas ao nível de tempo necessário para efetuar a redução do modelo3D, apenas necessita de poucos minutos (1-2 minutos) para que a simplificação seja efetuada.

O Simplygon ao simplificar tenta manter o detalhe nas zonas que apresentam maior detalhe (ou seja, tenta manter o rácio de triângulos nas zonas igual), uma vez que zonas com maior detalhe mantém maior número de triângulos/arestas em comparação a zonas de menor detalhe.

Quanto ao Open3D, apesar de ser muito mais rápido na execução da redução, este não tem em conta o detalhe e foca-se apenas em reduzir o número de triângulos através do aumento do seu tamanho e interpolação dos triângulos que foram agrupados.

Em ambos os casos foram efetuados reduções tendo em conta ao número de triângulos em 10% do original. Deste modo, o impacto de renderização no browser a nível de performance é semelhante, uma vez que o seu número de formas poligonais é também semelhante.

### 7.4.3 Impacto modelos 3D no browser

Esta experiência focou-se no impacto que os modelos 3D têm no browser quanto à sua performance. Um aspeto importante de analisar é o impacto que as reduções dos modelos têm para a sua performance/detalhes quando apresentados ao utilizador.

Foram então importados os modelos referenciados na Tabela 7-1 para o browser utilizando o PC1. Ao importar os modelos foi evidenciado desde o início que a performance dos modelos simplificados era bastante superior à dos modelos originalmente gerados.

Um teste efetuado ao modelo '*Bedroom0'* está apresentado na Figura 7-5, onde é possível verificar à esquerda o modelo sem ter sido simplificado e à direita o modelo simplificado no *Simplygon*.

O modelo simplificado apresenta um valor de *frames* por segundo estável nos 60, enquanto o modelo 3D não simplificado apresenta um valor máximo de 42 FPS.

Outro aspeto importante é o tempo que demora a ser carregado o modelo 3D para a cena. No teste apresentado anteriormente, o modelo não simplificado demorou cerca de 3 segundos a ser renderizado, enquanto o modelo que foi simplificado demora cerca de 350 milissegundos (o que é bastante inferior, ou seja, com muita melhor performance).

Além disto, a memória alocada à cena no modelo simplificado é no máximo 51MB enquanto no modelo não simplificado este valor é de 210 MB, o que implica uma alocação de memória superior.


Figura 7-5 – Comparação modelo não simplificado vs. simplificado

Os resultados obtidos da performance dos modelos 3D no browser estão discriminados na Tabela 7-2. Esta tabela contém informação quanto aos *frames* por segundo (FPS) obtidos aproximadamente, a memória alocada à cena (em Mega Bytes) e tempo de *loading* da cena.

Tabela 7-2 – Performance dos modelos em browser

| Modelo | FPS | Memória alocada (MB) | Tempo de *loading* (s) |
|---|---|---|---|
| **Bedroom0** | 42 | 516 | 2.9 |
| **Bedroom0 simplificado** | 60 | 51 | 0.315 |
| **Bedroom1** | 23 | 978 | 5.49 |
| **Bedroom1 simplificado** | 60 | 82 | 0.362 |
| **Bedroom2** | 22 | 668 | 3.20 |
| **Bedroom2 simplificado** | 60 | 59 | 0.742 |
| **OfficeHall** | 31 | 811 | 3.59 |
| **OfficeHall simplificado** | 60 | 26 | 0.727 |
| **LivingRoom** | 24 | 881 | 3.99 |
| **LivingRoom simplificado** | 60 | 29 | 0.763 |

Através dos resultados obtidos, é possível verificar a importância do processo de redução, uma vez que os modelos 3D simplificados apresentavam menor alocação de recursos e melhor desempenho ao nível da cena virtual num browser 3D, bem como no tempo de carregamento de cada caso.

## 7.5 Resultados obtidos

Após a análise das experiências efetuadas, foi concluído que:

- O Open3D permite recriar as *point clouds*, ainda que de uma forma um pouco pesada a nível computacional;
- O uso do *Simplygon* permite diminuir os modelos 3D de forma a reduzir o impacto dos modelos 3D gerados pelo Open3D;
- O processamento das *point cloud* e criação dos modelos 3D é um processo que necessita de recursos para agilizar e acelerar o processo de recriação dos modelos 3D.

Mais especificamente na utilização do Open3D:

- A reconstrução de *Poisson* é uma opção que foca mais o detalhe na reconstrução de interiores, em relação ao *Ball Pivoting Algorithm*;
- A reconstrução de *Poisson* é um processo demorado e o seu tempo de execução escala relativamente ao parâmetro *depth* definido, porém quanto maior o parâmetro, maior a qualidade do modelo 3D gerado (e maior também o seu peso computacional);
- Para o caso em específico foi definido um valor de 15 para a *depth*, como forma de relacionar os tempos/recursos necessários para a geração do modelo e a qualidade que este apresenta;
- O modelo 3D gerado pelo Open3D é passível de ser incorporado num website, porém é aconselhável a otimização do mesmo (neste caso, foi escolhida a utilização do *Simplygon* como ferramenta para o efetuar).

Quanto à utilização do *Simplygon*:

- Este ainda está disponibilizado apenas para o sistema operativo Windows;
- Permite efetuar reduções no modelo 3D sem ter um impacto considerável na qualidade do modelo 3D;
- Para o caso em específico, foi definido que efetuar a redução para 10% dos triângulos da malha já gera um modelo 3D final aceitável e passível de ser renderizado num website, de forma a ter uma boa performance.

Quanto à performance dos modelos num website:

- Quanto maior for o modelo, menor a performance obtida no website;
- Para minimizar o impacto do modelo 3D é necessário reduzir ao máximo possível os componentes utilizados para a criação das formas geométricas da malha;
- A utilização de *BackfaceCulling* quando os vetores normais estão disponíveis no modelo 3D e orientados corretamente é uma opção de renderização eficaz na redução de polígonos gerados na cena, o que aumenta a performance e a fluidez da mesma;
- Quanto maior for o espaço ocupado/tamanho pelo modelo 3D, maior será o tempo de renderização do modelo 3D na cena.

Os modelos 3D resultantes da execução da prova de conceito são apresentados na secção de Anexo – Experimentação e Avaliação.

# 8 Conclusões

Este capítulo tem como conteúdo as conclusões obtidas do projeto, constituído pelos objetivos alcançados, limitações e trabalho futuro. No final é apresentada uma apreciação geral ao projeto.

## 8.1 Objetivos alcançados

O objetivo do projeto visava a prova de que era possível recriar espaços interiores através do scan, ou seja, partindo de uma *point cloud* gerada por um utilizador.

Para alcançar o objetivo foram necessários resolver certos problemas encontrados ao longo do projeto, bem como problemas que surgiram na altura de pesquisa de informação. Problemas como:

- Definição da tecnologia a utilizar;
- Diversidade de *point clouds/modelos* bem como a forma de os representar;
- Necessidade de realizar o tratamento das *point clouds*;
- Criação de vetores normais aos pontos quando estes não existem;
- Comparar formas de recriação das *point clouds* em modelos 3D;
- Simplificação/Melhoria dos modelos 3D para que não fiquem demasiado pesados a nível computacional;
- Analisar os impactos dos modelos 3D num browser;
- Formas de melhorar a qualidade dos modelos 3D.

No final foram ainda respondidas as hipóteses colocadas na definição do Objetivo desta prova de conceito, sendo que:

- É possível recriar os interiores das habitações, partindo de scans das mesmas, e por sua vez, das *point cloud* geradas, sendo que é possível obter uma qualidade

considerável. Esta qualidade aumenta quanto maior forem os recursos disponibilizados para a reconstrução.

- Quanto à melhor estratégia para recriação dos espaços, existem várias estratégias e tecnologias para recriar a *point cloud*, porém estas devem ser adaptadas aos recursos e ao tipo de *point clouds*/reconstruções pretendidas. Porém, é possível identificar processos transversais, tais como: o tratamento e pré-processamento das *point clouds*, criação ou utilização de um algoritmo de reconstrução. Esta reconstrução pode ser ainda complementada com processos de melhoria do modelo gerado consoante o propósito desejado.

## 8.2 Limitações e trabalho futuro

Apesar de existir um pré-processamento da *point cloud*, quanto melhor for a qualidade do scan efetuado (*point cloud* com poucos outliers, sem bastante ruído, pouco fragmentada), melhor será o produto final (modelo 3D obtido). Este aspeto poderá levar a erros ou perda de qualidade no modelo 3D final.

Apesar de serem gerados os vetores normais dos pontos, caso não existam, este processo é sempre uma estimativa das normais dos pontos. Caso seja possível, é apropriado obter estes vetores normais através do processo de scan.

O detalhe do modelo 3D final pode ser melhorado, uma vez que é utilizada a cor dos pontos existentes para a renderização dos modelos. Este detalhe pode ser melhorado através da utilização de texturas e criação de mapeamento uv. Este é um fator de melhoria que pode ser implementado futuramente.

No futuro seria também interessante integrar o processo de recriação/criação dos modelos 3D aquando for feito o scan em vez de serem processo somente separados, sendo necessário o upload de uma *point cloud*. Desta forma, seria possível agilizar o processo.

Uma vez que o foco da prova de conceito se centra na digitalização de interiores de habitações, o próximo passo poderia passar pela inclusão do processo de recriação descrito neste documento, numa pipeline cujo objetivo final passaria por apresentar e guiar o utilizador pelo modelo, em estilo de uma visita virtual ao imóvel.

## 8.3 Apreciação final

Esta prova de conceito foi uma forma interessante de aprofundar conhecimentos na área de computação gráfica, mais especificamente na parte de manipulação de *point cloud* e formas de recriar/digitalizar os espaços do mundo real num ambiente virtual.

É expectável que este projeto, bem como a sua documentação e conclusões obtidas, sirva de suporte para a implementação de um mecanismo de reconstrução de scans efetuados por um

utilizador comum, uma possibilidade impulsionada pelo aumento da acessibilidade aos scanners (como é o caso da disponibilização de LiDAR no iPhone).

De uma forma geral o projeto foi cumprido e serviu como enriquecimento tanto a nível pessoal como a nível da empresa onde este projeto foi desenvolvido.

# Referências


[1] W. A. Abbasi e T. Ali, "Role of Augmented and Virtual Reality Marketing in Organizational Development.," *Journal of Marketing & Management,* pp. 1-19, 2020.

[2] O. Hasler, S. Blaser e S. Nebiker, "Performance Evaluation of a mobile mapping application using smartphones and augmented reality frameworks," *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.,* p. 741–747, 2020.

[3] A. S. Rodríguez, B. R. Rodríguez, M. S. Rodríguez e P. A. Sánchez, "Laser scanning and its applications to damage detection and monitoring in masonry structures," em *Long-term Performance and Durability of Masonry Structures*, B. Ghiassi e P. B. Lourenço, Edits., Woodhead Publishing, 2019, pp. 265-28.

[4] K. C. Chua, H. C. Wong e Y. W. Yeong, "Software and Data Format," em *Standards, Quality Control, and Measurement Sciences in 3D Printing and Additive Manufacturing*, K. C. Chua, H. C. Wong e Y. W. Yeong, Edits., Academic Press, 2017, pp. 75-94.

[5] M. Jan, "Pixels and voxels, the long answer," Medium, 26 09 2016. [Online]. Available: https://medium.com/retronator-magazine/pixels-and-voxels-the-long-answer-5889ecc18190. [Acedido em 02 12 2020].

[6] F. A. Merchant, K. A. Bartels, A. C. Bovik e K. R. Diller, "Confocal Microscopy," em *Handbook of Image and Video Processing*, 2ª ed., A. Bovik, Ed., Academic Press, 2005, pp. 1291-XLI.

[7] L. Kong e J. Hu, “A virtual 3D data exploration environment based on octree,” em *Proceedings 7th International Conference on Signal Processing*, Beijing, 2004.

[8] J. Liang e J. Gong, “A Sparse Voxel Octree-Based Framework for Computing Solar Radiation Using 3D City Models,” *International Journal of Geo-Information ,* vol. 6, nº 4, p. 106, 2017.

[9] A. Kanezaki, R. Kuga, Y. Sugano e Y. Matsushita, “Deep Learning for Multimodal Data Fusion,” em *Multimodal Scene Understanding*, M. Y. Yang, B. Rosenhahn e V. Murino, Edits., Academic Press, 2019, pp. 9-39.

[10] C. Zou, R. Guo, Z. Li e D. Hoiem, “Complete 3D Scene Parsing from an RGBD Image,” *International Journal of Computer Vision,* vol. 127, p. 143–162, 21 11 2018.

[11] J. Xiao e Y. Furukawa, “Reconstructing the world's museums,” *International Journal Computer Vision,* vol. 110, nº 3, pp. 243-258, 2014.

[12] R. Parent, “Modeling and Animating Human Figures,” em *Computer Animation (Third Edition)*, R. Parent, Ed., Morgan Kaufmann, 2012, pp. 283-315.

[13] B. Preim e C. Botha, “Virtual Endoscopy,” em *Visual Computing for Medicine (Second Edition)*, B. Preim e C. Botha, Edits., Morgan Kaufmann, 2014, pp. 509-536.

[14] P. Lamon, “3D-Odometry,” em *3D-Position Tracking and Control for All-Terrain Robots*, vol. 43, Berlin, Springer, 2008, pp. 21-32.

[15] K. Cao, X. Yang, S. Gao, C. Chen, J. Huang e X. Song, “Visual Odometry Based on 3D-3D and 3D-2D Motion Estimation Method,” em *2018 Chinese Automation Congress (CAC)*, Xi'an, China, 2018.

[16] A. Mordvintsev e A. K., “ORB (Oriented FAST and Rotated BRIEF),” OpenCV, 2013. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_orb/py_orb.html. [Acedido em 11 12 2020].

[17] Vision Online Marketing Team, “What is Visual SLAM Technology and What is it Used For?,” Global Association for Visual Information, 15 05 2018. [Online]. Available: https://www.visiononline.org/blog-article.cfm/What-is-Visual-SLAM-Technology-and-What-is-it-Used-For/99. [Acedido em 5 12 2020].

[18] S. Bala, “Introducing SLAM,” Arm Community, 13 09 2018. [Online]. Available: https://community.arm.com/developer/tools-software/graphics/b/blog/posts/introducing-slam-technology. [Acedido em 5 12 2020].

[19] J. Castagno e E. Atkins, “Polylidar3D-Fast Polygon Extraction from 3D Data,” *Sensors,* nº 17: 4819, 2020.

[20] Y. Cui, Q. Li, B. Yang, W. Xiao, C. Chen e Z. Dong, “Automatic 3-D Reconstruction of Indoor Environment With Mobile Laser Scanning Point Clouds,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 12, nº 8, pp. 3117 - 3130, 2019.

[21] C. Wang, S. Hou, C. Wen, Z. Gong, Q. Li, X. Sun e J. Li, “Semantic line framework-based indoor building modeling using backpacked laser scanning point cloud,” *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 143, pp. 150-166, 2018.

[22] L. Díaz-Vilariño, P. Boguslawski, K. Khoshelham, H. Lorenzo e L. Mahdjoubi, “Indoor navigation from point clouds: 3D modelling and obstacle detection,” *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci,* Vols. %1 de %2XLI-B4, pp. 275-281, 2016.

[23] Y. Lin, C. Wang, B. Chen, D. Zai e J. Li, “Facet segmentation-based line segment extraction for large-scale point clouds,” *IEEE Trans. Geosci. Remote Sens,* vol. 55, nº 9, pp. 4839-4854, 2017.

[24] C. Liu, J. Wu e Y. Furukawa, “FloorNet: A unified framework for floorplan reconstruction from 3D scans,” em *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[25] V. Sanchez e A. Zakhor, “Planar 3D modeling of building interiors from point cloud data,” em *Proc. IEEE Int. Conf. Image Processing*, 2013.

[26] S. Oesau, F. Lafarge e P. Alliez, “Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut,” *ISPRS J. Photogramm. Remote Senses,* vol. 90, pp. 68-82, 2014.

[27] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva e G. Taubin, “The ball-pivoting algorithm for surface reconstruction,” *IEEE Transactions on Visualization and Computer Graphics,* vol. 5, nº 4, pp. 349-359, 1999.

[28] F. Poux, “5-Step Guide to generate 3D meshes from point clouds with Python,” Medium, 21 04 2020. [Online]. Available: https://towardsdatascience.com/5-step-guide-to-generate-3d-meshes-from-point-clouds-with-python-36bad397d8ba. [Acedido em 2 12 2020].

[29] M. Kazhdan, M. Bolitho e H. Hoppe, “Poisson Surface Reconstruction,” em *Eurographics Symposium on Geometry Processing*, 2006.

[30] N. Qian, “Efficient Poisson-Based Surface Reconstruction of 3D Model from a Non-homogenous Sparse Point Cloud,” em *Image and Signal Processing*, Springer, Cham, 2014, pp. 578-585.

[31] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys e A. Geiger, “Convolutional Occupancy Networks,” em *European Conference on Computer Vision (ECCV)*, 2020.

[32] Google, “ARCore overview,” 06 11 2020. [Online]. Available: https://developers.google.com/ar/discover.

[33] Google, “Fundamental concepts,” Google, 06 11 2020. [Online]. Available: https://developers.google.com/ar/discover/concepts. [Acedido em 15 11 2020].

[34] H. H. K. Niazi, “Ultimate Guide to Augmented Reality,” Medium, 23 12 2018. [Online]. Available: https://medium.com/@it.hhkn/ultimate-guide-to-augmented-reality-83c9d511d842. [Acedido em 10 11 2020].

[35] American Geosciences Institute, “What is Lidar and what is it used for?,” American Geosciences Institute, 2020. [Online]. Available: https://www.americangeosciences.org/critical-issues/faq/what-lidar-and-what-it-used. [Acedido em 20 11 2020].

[36] ArcGIS Desktop, “Types of Lidar,” ArcGIS Desktop, 2019. [Online]. Available: https://desktop.arcgis.com/en/arcmap/10.3/manage-data/las-dataset/types-of-lidar.htm. [Acedido em 20 11 2020].

[37] B. Sharma, “What is LiDAR technology and how does it work?,” Geospatial Worlds, 16 10 2020. [Online]. Available: https://www.geospatialworld.net/blogs/what-is-lidar-technology-and-how-does-it-work/. [Acedido em 20 11 2020].

[38] Matterport, Inc., “How Matterport works,” Matterport, Inc., 2020. [Online]. Available: https://matterport.com/how-it-works. [Acedido em 10 11 2020].

[39] Polycam, “Polycam,” Polycamai, [Online]. Available: https://poly.cam/. [Acedido em 24 02 2021].

[40] Occipital Inc., “Canvas by occipital,” Occipital Inc., 2020. [Online]. Available: https://canvas.io/. [Acedido em 27 02 2021].

[41] R. B. Rusu e S. Cousins, “3D is here: Point Cloud Library (PCL),” *IEEE International Conference on Robotics and Automation 2011 (ICRA 2011),* 05 2011.

[42] Q.-Y. Zhou, V. Koltun e J. Park, “Open3D: A Modern Library for 3D Data Processing,” *arXiv:1801.09847,* 2018.

[43] T. Chaton, N. Chaulet, S. Horache e L. Landrieu, “Torch-Points3D: A Modular Multi-Task Frameworkfor Reproducible Deep Learning on 3D Point Clouds,” em *International Conference on 3D Vision (3DV)*, 2020 .

[44] Simplygon, “Simplygon 9 Documentation,” Microsoft, 2020. [Online]. Available: https://www.simplygon.com/. [Acedido em 12 03 2021].

[45] ThreeJS, “Three.js Fundamentals,” [Online]. Available: https://threejsfundamentals.org/threejs/lessons/threejs-fundamentals.html#toc. [Acedido em 02 12 2020].

[46] S. Nicola, “Análise_Valor_Aula_1_18NOV,” 2020.

[47] S. Nicola, “Análise_Valor_Aula_2_25NOV,” 2020.

[48] P. Koen, G. Ajamin, R. Burkart, A. Clamen, J. Davidson, R. D'Amore, C. Elkins, K. Herald, M. Incorvia e A. Johnson, “Providing clarity and a common language to,” *Research-Technology Management,* vol. 44, nº 2, p. 46–55, 2001.

[49] N. Fain e B. Wagner, “Management of the fuzzy front end of innovation, by Oliver,” *International Journal of Market Research,* vol. 58, p. 637, 2016.

[50] A. Osterwalder e Y. Pigneur, “Modeling value propositions in e-Business,” em *Proceedings of the 5th international conference on Electronic commerce*, 2003.

[51] J. A. Rains, *What are the Functions of Function Analysis,* Arizona, USA: Advanced Value Group, LLC, 2009.

[52] M. Beynon, “DS/AHP method: A mathematical analysis, including an understanding of uncertainty,” *European Journal of Operational Research,* vol. 140, nº 1, pp. 148-164, 2002.

[53] P. B. Kruchten, “ The 4+ 1 view model of architecture,” *IEEE software,* vol. 12, nº 6, p. 42–50, 1995.

[54] P. Sapra, “Kruchten’s 4 + 1 views of Software Design,” Medium, 26 03 2018. [Online]. Available: https://medium.com/the-mighty-programmer/kruchtens-views-of-software-design-e9088398c592. [Acedido em 16 01 2021].

[55] I. Sommerville, “Chapter 8 - Software testing,” em *Software Engineering*, 10th ed., Londres, Pearson, 2016, pp. 205-233.

[56] R. W. Miller e C. T. Collins, “Acceptance Testing,” *Proc. XPUniverse,* p. 238, 2001.

[57] D. Odhiambo, “Performance Optimization in Software Development,” 24 09 2018. [Online]. Available: https://medium.com/the-andela-way/performance-optimization-in-software-development-ae7952ab885e. [Acedido em 10 01 2021].

[58] mrdoob, “stats.js,” 1 2021. [Online]. Available: https://github.com/mrdoob/stats.js/. [Acedido em 04 04 2021].

[59] P. Cloud, “https://pypi.org/project/axil-autotime/,” 07 10 2019. [Online]. Available: https://pypi.org/project/axil-autotime/. [Acedido em 14 05 2021].

[60] Apple;, “Introducing ARKit 4,” 2020. [Online]. Available: https://developer.apple.com/augmented-reality/arkit/.

[61] N. Rich e M. Holweg, “Value engineering: Innoregio: dissemination of innovation and knowledge management techniques,” 2000.

[62] D. Hughes e D. Chafin, “Turning New Product Development into a Continuous Learning”.

[63] Google VR, “Degrees of freedom,” Google, 21 09 2018. [Online]. Available: https://developers.google.com/vr/discover/degrees-of-freedom. [Acedido em 10 12 2020].

[64] D. Tyagi, “Introduction to FAST (Features from Accelerated Segment Test),” Medium, 02 01 2019. [Online]. Available: https://medium.com/data-breach/introduction-to-fast-features-from-accelerated-segment-test-4ed33dde6d65. [Acedido em 12 11 2020].

[65] D. Tyagi, “Introduction to BRIEF(Binary Robust Independent Elementary Features),” Medium, 19 03 2019. [Online]. Available: https://medium.com/data-breach/introduction-to-brief-binary-robust-independent-elementary-features-436f4a31a0e6. [Acedido em 12 11 2020].

[66] Y. Xiang e D. Fox, “DA-RNN: Semantic Mapping with Data Associated Recurrent Neural Networks,” *CoRR,* vol. 1703.03098, 2017.

# A. Anexo – Método AHP

O método AHP foi utilizado para encontrar a tecnologia que melhor se adequava às necessidades

Tabela A-0-1 - Valores de IR para matrizes quadradas de ordem n [47]

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.00 | 0.00 | 0.58 | 0.90 | 1.12 | 1.24 | 1.32 | 1.41 | 1.45 | 1.49 | 1.51 | 1.48 | 1.56 | 1.57 | 1.59 |

De seguida é apresentado como foi implementado este método.

# A.1 Implementação método AHP

## A.1.1 Fase 1

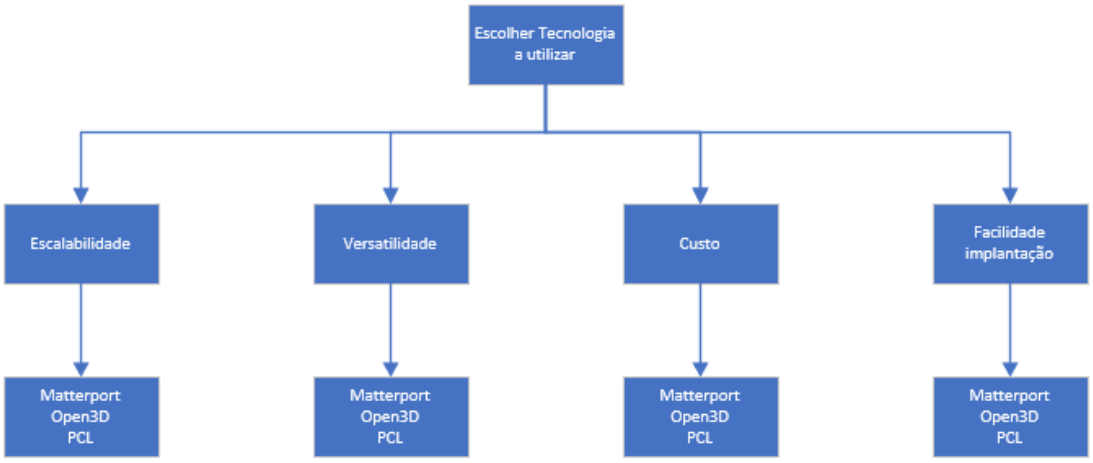


Figura A.0-1 – Árvore hierárquica de decisão

## A.1.2 Fase 2

Nesta fase foi realizada a comparação entre os critérios da hierarquia (descritos no parágrafo 5.1 e na Figura A.0-1), obtendo a seguinte comparação (Tabela A-0-2).

Tabela A-0-2 – Comparação dos critérios do AHP

| Comparação elementos | | | | |
|---|---|---|---|---|
| | Escalabilidade | Versatilidade | Custo | Facilidade implantação |
| Escalabilidade | 1 | 5 | 1/4 | 4 |
| Versatilidade | 1/5 | 1 | 1/7 | 2 |
| Custo | 4 | 7 | 1 | 6 |
| Facilidade implantação | 1/4 | 1/2 | 1/6 | 1 |
| Soma | 49/9 | 27/2 | 14/9 | 13/1 |

## A.1.3 Fase 3

De seguida foram normalizados os valores da matriz de comparação (Tabela A-0-3), como forma de obter o vetor de prioridades relativas, obtendo assim o peso de cada critério.

Tabela A-0-3 – Matriz normalizada dos critérios de seleção

| Matriz normalizada critérios | | | | | |
|---|---|---|---|---|---|
| | Escalabilidade | Versatilidade | Custo | Facilidade implantação | Prioridade Relativa |
| Escalabilidade | 11/60 | 10/27 | 4/25 | 4/13 | 0,26 |
| Versatilidade | 1/27 | 2/27 | 1/11 | 2/13 | 0,09 |
| Custo | 11/15 | 14/27 | 25/39 | 6/13 | 0,59 |
| Facilidade implantação | 1/22 | 1/27 | 3/28 | 1/13 | 0,07 |

Desta forma é possível averiguar que:

- Escalabilidade tem uma prioridade relativa de 0.26;
- Versatilidade tem uma prioridade relativa de 0.09;
- Custo tem uma prioridade relativa de 0.59;
- Facilidade de implantação tem uma prioridade relativa de 0.07.

### A.1.4 Fase 4

Nesta fase é avaliada a consistência das prioridades relativas encontradas. Para que estas sejam consistentes o Rácio de Consistência (RC) tem de ser menor que 0.1.

Para efetuar o cálculo do RC: $RC = IC/0.9$, onde IC é o Índice de Consistência e 0.9 é o valor obtido pelos valores tablados em Tabela A-0-1 para ordem 4 (4 critérios).

O IC é obtido através de: $IC = \frac{\lambda max - n}{n-1}, sendo\ que\ \lambda max.x = A.x$ onde A é a matriz normalizada e x o vetor das prioridades relativas.

Assim sendo:

$$\begin{bmatrix} 1 & 5 & \frac{1}{4} & 4 \\ \frac{1}{5} & 1 & \frac{1}{7} & 2 \\ 4 & 7 & 1 & 6 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{6} & 1 \end{bmatrix} \begin{bmatrix} 0.26 \\ 0.09 \\ 0.59 \\ 0.07 \end{bmatrix} \cong \lambda max. \begin{bmatrix} 0.26 \\ 0.09 \\ 0.59 \\ 0.07 \end{bmatrix} \Leftrightarrow \begin{bmatrix} 1.11 \\ 0.36 \\ 2.63 \\ 0.27 \end{bmatrix} \cong \lambda max. \begin{bmatrix} 0.26 \\ 0.09 \\ 0.59 \\ 0.07 \end{bmatrix}$$

$$\lambda \max \cong média \left\{ \frac{1.11}{0.26}, \frac{0.36}{0.09}, \frac{2.63}{0.59}, \frac{0.27}{0.07} \right\} \cong 4.24$$

Assim sendo, o IC é de:

$$IC = \frac{(4.24 - 4)}{4 - 1} \cong 0.08$$

Desta forma, é possível agora calcular o RC de 0.09:

$$RC = \frac{0.08}{0.9} \cong 0.09$$

Uma vez que o RC < 0.1, é possível aferir que existe consistência dos critérios.

### A.1.5 Fase 5

Após a confirmação de consistência dos critérios, são construídas as matrizes de comparação paritária dos critérios para cada uma das tecnologias a avaliar. Estas matrizes são apresentadas nas tabelas seguintes:

Tabela A-0-4 – Matriz paritária para o critério de escalabilidade

| Critério: Escalabilidade | | | | |
|---|---|---|---|---|
| | Matterport | PCL | Open3D | Prioridade Relativa |
| Matterport | 1 | 5 | 4 | 3,33 |
| PCL | 1/5 | 1 | 1/2 | 0,57 |
| Open3D | 1/4 | 2 | 1 | 1,08 |

Tabela A-0-5 - Matriz paritária para o critério de versatilidade

| Critério: Versatilidade | | | | |
|---|---|---|---|---|
| | Matterport | PCL | Open3D | Prioridade Relativa |
| Matterport | 1 | 5 | 4 | 3,33 |
| PCL | 1/5 | 1 | 1/3 | 0,51 |
| Open3D | 1/4 | 3 | 1 | 1,42 |

Tabela A-0-6 - Matriz paritária para o critério de custo

| Critério: Custo | | | | |
|---|---|---|---|---|
| | Matterport | PCL | Open3D | Prioridade Relativa |
| Matterport | 1 | 1/7 | 1/7 | 0,43 |
| PCL | 7 | 1 | 1 | 3,00 |
| Open3D | 7 | 1 | 1 | 3,00 |

Tabela A-0-7 - Matriz paritária para o critério de facilidade implantação

| Critério: Facilidade implantação | | | | |
|---|---|---|---|---|
| | Matterport | PCL | Open3D | Prioridade Relativa |
| Matterport | 1 | 2 | 4 | 2,33 |
| PCL | 1/2 | 1 | 2 | 1,17 |
| Open3D | 1/4 | 1/2 | 1 | 0,58 |

### A.1.6 Fase 6

Nesta fase é criada a matriz de prioridades compostas para cada uma das alternativas. Nesta fase é criada uma matriz prioridade das alternativas analisadas e feita o produto com a matriz dos pesos dos critérios (encontrados na Fase 3).

$$\begin{matrix} Matterport \\ PCL \\ \boldsymbol{Open3D} \end{matrix} \begin{bmatrix} 3.33 & 3.33 & 0.43 & 2.33 \\ 0.57 & 0.51 & 3.00 & 1.17 \\ 1.08 & 1.42 & 3.00 & 0.58 \end{bmatrix} . \begin{bmatrix} 0.26 \\ 0.09 \\ 0.59 \\ 0.07 \end{bmatrix} = \begin{bmatrix} 1.56 \\ 2.03 \\ \mathbf{2.21} \end{bmatrix}$$

### A.1.7 Fase 7

Nesta fase é realizada a escolha da melhor tecnologia a utilizar para o caso, tendo por base a matriz de prioridades compostas, criada na fase anterior. Assim sendo, a tecnologia que melhor se adequa à situação é o Open3D.

# B. Anexo – Experimentação e Avaliação

## B.1 Tempo de execução criação do modelo 3D no PC1

Os tempos de execução em segundos estão presentes nas tabelas abaixo (são referentes à *point cloud* da BedRoom0, com 1 000 000 de pontos iniciais) onde se referem a utilização da *point cloud* com o número inicial, com *downsample* de 0.03 (resulta em 49 209 pontos) e com *downsample* de 0.5 (resulta em 18 581 pontos).

Tabela B-0-8 – Tempos execução PC1 sem *downsample*

| Downsample (-) | Pontos inicias: 1 000 000 | | | Voxelized: 988 550 | | | Mesh Triangles: 3 502 998 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pre - Process Cloud | | | Reconstruction methods && mesh improvement methods | | | | | |
| | Downsample method | Estimate normals | Outlier Remove | Poisson mesh (depth15) | Decimal simplification 70% | Decimal simplification 50% | Decimal simplification 10% | Total (sec) | Total (min) |
| | 0.00 | 6.33 | 4.12 | 383.00 | 63.00 | 79.00 | 121.00 | 652.33 | 10:52 |
| | 0.00 | 5.78 | 4.10 | 371.23 | 60.00 | 74.00 | 117.00 | 628.01 | 10:28 |
| | 0.00 | 5.88 | 4.15 | 368.87 | 65.00 | 76.00 | 119.00 | 634.75 | 10:35 |
| | 0.00 | 5.74 | 4.18 | 376.95 | 59.00 | 72.00 | 113.00 | 626.69 | 10:27 |
| | 0.00 | 5.75 | 4.11 | 373.56 | 62.00 | 77.00 | 115.00 | 633.31 | 10:33 |
| Mean | 0.00 | 5.90 | 4.13 | 374.72 | 61.80 | 75.60 | 117.00 | 635.02 | 10:35 |
| Min | 0.00 | 5.74 | 4.10 | 368.87 | 59.00 | 72.00 | 113.00 | 618.61 | 10:19 |
| Max | 0.00 | 6.33 | 4.18 | 383.00 | 65.00 | 79.00 | 121.00 | 654.33 | 10:54 |

Tabela B-0-9 - Tempos execução PC1 com *downsample* com *voxelsize* de 0.03

| Downsample (0.03) | Pontos inicias: 1 000 000 | | | Voxelized: 49 209 | | | Mesh Triangles: 1 906 247 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pre - Process Cloud | | | Reconstruction methods && mesh improvement methods | | | | | |
| | Downsample method | Estimate normals | Outlier Remove | Poisson mesh (depth15) | Decimal simplification 70% | Decimal simplification 50% | Decimal simplification 10% | Total (sec) | Total (min) |
| | 1.41 | 2.19 | 3.35 | 174.00 | 27.10 | 32.10 | 47.10 | 283.90 | 04:44 |
| | 1.10 | 2.34 | 2.96 | 173.40 | 24.40 | 34.20 | 46.20 | 281.64 | 04:42 |
| | 1.25 | 1.56 | 3.12 | 174.50 | 24.50 | 36.20 | 46.70 | 284.71 | 04:45 |
| | 1.10 | 1.70 | 3.15 | 174.90 | 24.10 | 34.50 | 45.80 | 282.10 | 04:42 |
| | 1.07 | 1.40 | 3.10 | 174.30 | 24.30 | 32.89 | 47.00 | 280.96 | 04:41 |
| Mean | 1.19 | 1.84 | 3.14 | 174.40 | 24.88 | 33.98 | 46.56 | 282.84 | 04:43 |
| Min | 1.07 | 1.40 | 2.96 | 173.40 | 24.10 | 32.10 | 45.80 | 277.87 | 04:38 |
| Max | 1.41 | 2.34 | 3.35 | 174.90 | 27.10 | 36.20 | 47.10 | 289.05 | 04:49 |

Tabela B-0-10 – Tempos execução PC1 com *downsample* com *voxelsize* de 0.05

| Downsample (0.05) | Pontos inicias: 1 000 000 | | | Voxelized: 18 581 | | | Mesh Triangles: 610 897 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pre - Process Cloud | | | Reconstruction methods && mesh improvement methods | | | | | |
| | Downsample method | Estimate normals | Outlier Remove | Poisson mesh (depth15) | Decimal simplification 70% | Decimal simplification 50% | Decimal simplification 10% | Total (sec) | Total (min) |
| | 2.12 | 1.12 | 1.09 | 66.50 | 4.10 | 6.70 | 10.60 | 91.14 | 01:31 |
| | 2.01 | 1.23 | 1.25 | 65.30 | 4.20 | 6.80 | 9.50 | 89.04 | 01:29 |
| | 2.06 | 1.20 | 0.98 | 66.30 | 4.80 | 7.00 | 9.80 | 91.16 | 01:31 |
| | 2.23 | 1.17 | 1.13 | 67.10 | 4.60 | 6.40 | 10.30 | 91.80 | 01:32 |
| | 2.18 | 1.13 | 0.94 | 65.70 | 5.00 | 6.70 | 10.10 | 90.81 | 01:31 |
| Mean | 2.12 | 1.17 | 1.08 | 66.18 | 4.54 | 6.72 | 10.06 | 90.79 | 01:31 |
| Min | 2.01 | 1.12 | 0.94 | 65.30 | 4.10 | 6.40 | 9.50 | 88.43 | 01:28 |
| Max | 2.23 | 1.23 | 1.25 | 67.10 | 5.00 | 7.00 | 10.60 | 93.16 | 01:33 |

## B.2. Tempo de execução criação do modelo 3D no PC2

Os tempos de execução em segundos estão presentes nas tabelas acima (são referentes à *point cloud* da BedRoom0, com 1 000 000 de pontos iniciais) onde se referem a utilização da *point cloud* com o número inicial, com *downsample* de 0.03 (resulta em 49 209 pontos) e com *downsample* de 0.5 (resulta em 18 581 pontos).

Tabela B-0-11 – Tempos execução PC1 sem *downsample*

| Downsample (-) | Pontos inicias: 1 000 000 | | | Voxelized: 988 550 | | | Mesh Triangles: 3 502 998 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pre - Process Cloud | | | Reconstruction methods && mesh improvement methods | | | | | |
| | Downsample method | Estimate normals | Outlier Remove | Poisson mesh (depth15) | Decimal simplification 70% | Decimal simplification 50% | Decimal simplification 10% | Total (sec) | Total (min) |
| | 0.00 | 5.12 | 3.15 | 129.52 | 41.20 | 58.11 | 61.30 | 295.25 | 04:55 |
| | 0.00 | 5.26 | 3.24 | 126.65 | 42.01 | 58.90 | 61.42 | 294.24 | 04:54 |
| | 0.00 | 5.10 | 3.16 | 130.52 | 41.80 | 57.67 | 60.95 | 296.04 | 04:56 |
| | 0.00 | 4.78 | 3.02 | 129.32 | 40.90 | 58.10 | 61.12 | 294.22 | 04:54 |
| | 0.00 | 5.23 | 3.09 | 128.42 | 41.56 | 58.03 | 60.78 | 294.02 | 04:54 |
| Mean | 0.00 | 5.10 | 3.13 | 128.89 | 41.49 | 58.16 | 61.11 | 294.75 | 04:55 |
| Min | 0.00 | 4.78 | 3.02 | 126.65 | 40.90 | 57.67 | 60.78 | 290.78 | 04:51 |
| Max | 0.00 | 5.26 | 3.24 | 130.52 | 42.01 | 58.90 | 61.42 | 298.11 | 04:58 |

Tabela B-0-12 - Tempos execução PC1 com *downsample* com *voxelsize* de 0.03

| Downsample (0.03) | Pontos inicias: 1 000 000 | | | Voxelized: 49 209 | | | Mesh Triangles: 1 906 247 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Pre - Process Cloud | | | Reconstruction methods && mesh improvement methods | | | | | |
| | Downsample method | Estimate normals | Outlier Remove | Poisson mesh (depth15) | Decimal simplification 70% | Decimal simplification 50% | Decimal simplification 10% | Total (sec) | Total (min) |
| | 1.06 | 2.14 | 2.86 | 90.42 | 16.87 | 23.14 | 34.15 | 167.78 | 02:48 |
| | 1.01 | 2.10 | 2.77 | 91.32 | 16.12 | 23.01 | 34.11 | 167.67 | 02:48 |
| | 0.88 | 2.05 | 2.80 | 94.12 | 16.54 | 23.12 | 34.10 | 170.81 | 02:51 |
| | 1.12 | 2.06 | 2.78 | 91.90 | 15.96 | 22.99 | 33.94 | 167.97 | 02:48 |
| | 1.08 | 2.10 | 2.90 | 90.85 | 16.98 | 23.10 | 24.19 | 158.30 | 02:38 |
| Mean | 1.03 | 2.09 | 2.82 | 174.40 | 16.49 | 23.07 | 32.10 | 249.18 | 04:09 |
| Min | 0.88 | 2.05 | 2.77 | 90.42 | 15.96 | 22.99 | 24.19 | 156.49 | 02:36 |
| Max | 1.12 | 2.14 | 2.90 | 94.12 | 16.98 | 23.14 | 34.15 | 171.65 | 02:52 |

Tabela 0-13 – Tempos execução PC1 com *downsample* com *voxelsize* de 0.05

| Downsample (0.05) | Pontos inicias: 1 000 000 | | | Voxelized: 18 581 | | | Mesh Triangles: 610 897 | |
|---|---|---|---|---|---|---|---|---|
| | Pre - Process Cloud | | | Reconstruction methods && mesh improvement methods | | | | |
| | Downsample method | Estimate normals | Outlier Remove | Poisson mesh (depth15) | Decimal simplification 70% | Decimal simplification 50% | Decimal simplification 10% | Total (sec) | Total (min) |
| | 1.25 | 1.35 | 1.12 | 45.12 | 5.43 | 7.62 | 9.12 | 69.89 | 01:10 |
| | 1.24 | 1.45 | 1.14 | 46.32 | 5.77 | 7.40 | 9.98 | 72.16 | 01:12 |
| | 1.12 | 1.34 | 1.08 | 44.13 | 6.34 | 7.34 | 10.12 | 70.39 | 01:10 |
| | 1.20 | 1.39 | 1.05 | 45.36 | 5.23 | 7.23 | 10.01 | 70.42 | 01:10 |
| | 1.18 | 1.40 | 1.12 | 44.91 | 5.44 | 7.65 | 9.67 | 70.25 | 01:10 |
| Mean | 1.20 | 1.39 | 1.10 | 45.17 | 5.64 | 7.45 | 9.78 | 70.62 | 01:11 |
| Min | 1.12 | 1.34 | 1.05 | 44.13 | 5.23 | 7.23 | 9.12 | 68.17 | 01:08 |
| Max | 1.25 | 1.45 | 1.14 | 46.32 | 6.34 | 7.65 | 10.12 | 73.13 | 01:13 |

## B.3. Tempos de carregamento no browser

Nas figuras seguintes são apresentados os valores das experiências dos tempos necessários para os modelos 3D serem carregados para o browser. Os modelos cujo nome termina em 0_01 são as versões simplificadas dos modelos cujo nome terminam em GLB.

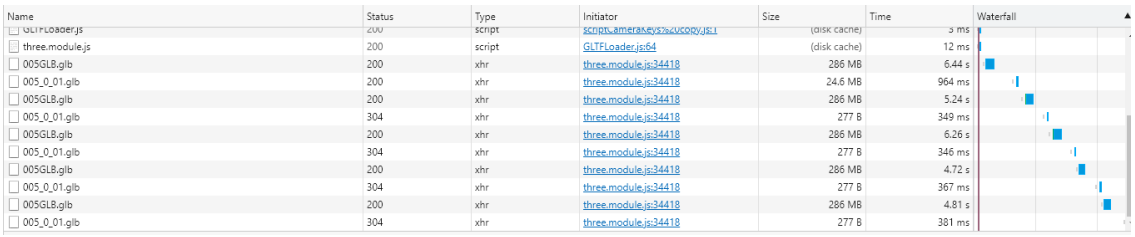


Figura 0-2 – Tempo de carregamento para o browser do modelo 3D 'Bedroom1'

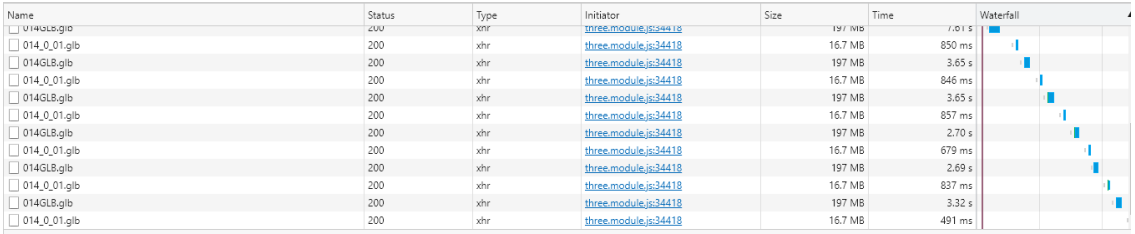


Figura 0-3 – Tempo de carregamento para o browser do modelo 3D 'Bedroom2'

| Name | Status | Type | Initiator | Size | Time | Waterfall |
|---|---|---|---|---|---|---|
| 011GLB.glb | 200 | xhr | three.module.js:34418 | 183 MB | 4.41 s | |
| 011_0_01.glb | 200 | xhr | three.module.js:34418 | 15.8 MB | 1.20 s | |
| 011GLB.glb | 200 | xhr | three.module.js:34418 | 183 MB | 3.81 s | |
| 011_0_01.glb | 200 | xhr | three.module.js:34418 | 15.8 MB | 553 ms | |
| 011GLB.glb | 200 | xhr | three.module.js:34418 | 183 MB | 3.28 s | |
| 011_0_01.glb | 200 | xhr | three.module.js:34418 | 15.8 MB | 808 ms | |
| 011GLB.glb | 200 | xhr | three.module.js:34418 | 183 MB | 3.20 s | |
| 011_0_01.glb | 200 | xhr | three.module.js:34418 | 15.8 MB | 555 ms | |
| 011GLB.glb | 200 | xhr | three.module.js:34418 | 183 MB | 3.28 s | |

Figura 0-4 – Tempo de carregamento para o browser do modelo 3D 'OfficeHall'

| Name | Status | Type | Initiator | Size | Time | Waterfall |
|---|---|---|---|---|---|---|
| 015GLB.glb | 200 | xhr | three.module.js:34418 | 199 MB | 4.15 s | |
| 015_0_01.glb | 200 | xhr | three.module.js:34418 | 18.2 MB | 1.08 s | |
| 015GLB.glb | 200 | xhr | three.module.js:34418 | 199 MB | 3.76 s | |
| 015_0_01.glb | 200 | xhr | three.module.js:34418 | 18.2 MB | 601 ms | |
| 015GLB.glb | 200 | xhr | three.module.js:34418 | 199 MB | 6.01 s | |
| 015_0_01.glb | 200 | xhr | three.module.js:34418 | 18.2 MB | 604 ms | |
| 015GLB.glb | 200 | xhr | three.module.js:34418 | 199 MB | 3.57 s | |
| 015_0_01.glb | 200 | xhr | three.module.js:34418 | 18.2 MB | 980 ms | |
| 015GLB.glb | 200 | xhr | three.module.js:34418 | 199 MB | 3.50 s | |
| 015_0_01.glb | 200 | xhr | three.module.js:34418 | 18.2 MB | 550 ms | |

Figura 0-5 – Tempo de carregamento para o browser do modelo 3D 'LivingRoom'

## B.4. Modelos 3D finais e visualizados no browser

Nas seguintes figuras são apresentados os modelos 3D simplificados e visualizados através do browser.

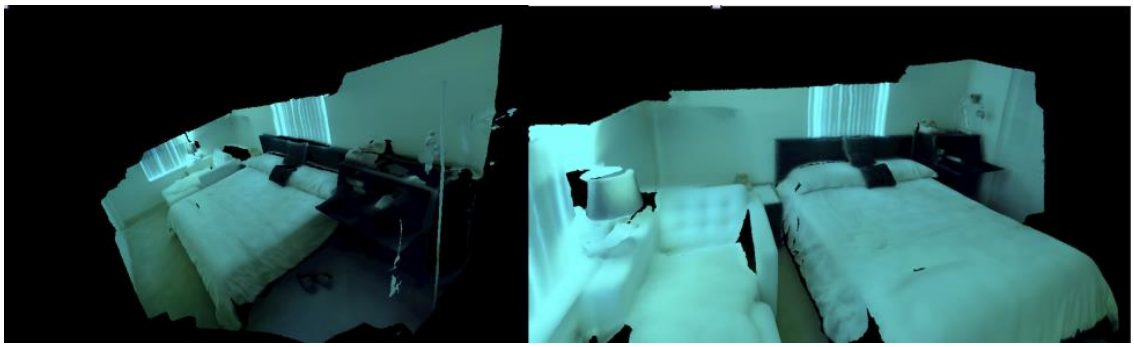

Figura 0-6 – modelo 'Bedroom0' visualizada num browser

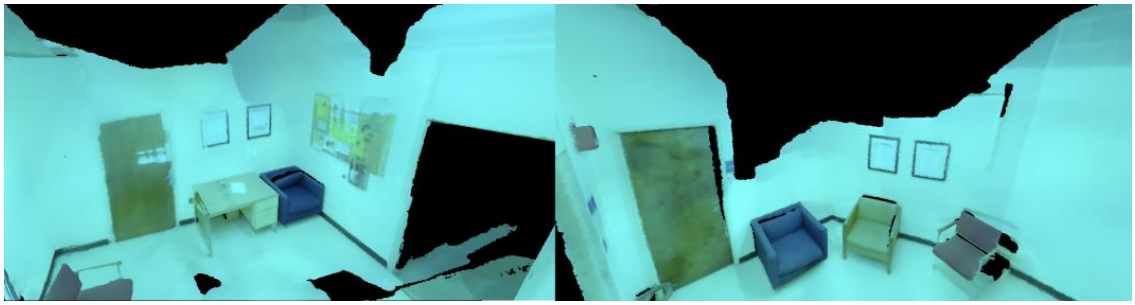

Figura 0-7 – modelo 'OfficeHall' visualizada num browser

Figura 0-8 – modelo ‘Bedroom1’ visualizado no browser

Figura 0-9 – modelo ‘Bedroom2’ visualizado no browser

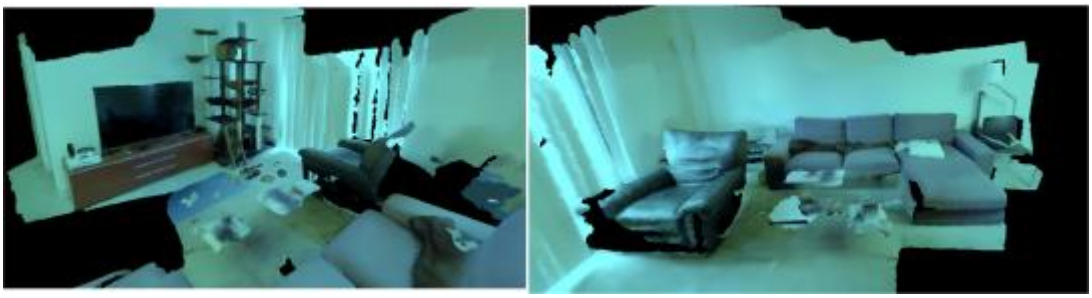

Figura 0-10 – modelo ‘LivingRoom’ visualizado no browser