UNIVERSIDADE FEDERAL DO PAMPA

Guilherme Neri Bustamante Sá

Um Método Escalável para Recuperação de Objetos 3D Utilizando Árvore de Vocabulário

Alegrete
2019

Guilherme Neri Bustamante Sá

Um Método Escalável para Recuperação de Objetos 3D Utilizando Árvore de Vocabulário

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Marcelo Resende Thielo

Alegrete
2019

Guilherme Neri Bustamante Sá

Um Método Escalável para Recuperação de Objetos 3D Utilizando Árvore de Vocabulário

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em 29 de novembro de 2019
Banca examinadora:

Prof. Doutor Marcelo Resende Thielo
Orientador
UNIPAMPA

Prof. Dra. Alice Fonseca Finger
UNIPAMPA

Prof. Mc. Eliezer Soares Flores
UNIPAMPA

Ao meus pais que, durante toda minha vida,
fizeram o possível e o impossível para que
eu chegasse a esta etapa da minha vida.

AGRADECIMENTOS

Agradeço e dedico este trabalho especialmente aos meus pais e irmã, que me apoiaram nos momentos mais difíceis dessa jornada. Não vou esquecer os muitos momentos em que um simples abraço me deu forças para levantar e continuar lutando. Não dedico somente este trabalho à vocês, mas sim tudo. Sem vocês nada disso seria possível. Eu sempre vou amar vocês.

Também agradeço aos amigos que fizeram parte dessa jornada. Aos que infelizmente não concluíram esta fase junto comigo, mas especialmente aos que me deram incentivo para concluir.

Agradeço especialmente aos amigos Lucas Antunes, Rafael Fernandes e Felipe Homrich que me mostraram como a vida pode ser cheia de surpresas. Embora tenha sido por um curto período, vocês me acolheram como irmãos, compartilhando as noites sem dormir, muitas vezes por conta de trabalhos que pareciam impossíveis, mas que com o trabalho em equipe puderam ser vencidos.

Por fim agradeço ao professor Marcelo Thielo pela orientação, amizade e por desempenhar o papel de professor com maestria. Sem dúvidas o senhor foi parte fundamental em minha graduação e vida.

"Você pode encontrar as coisas que perdeu,
mas nunca as que abandonou."
(J. R. R. Tolkien)

RESUMO

Interpretar uma cena, do ponto de vista computacional, geralmente se resume a interpretar e classificar seus elementos, percebendo como eles estão organizados pelo cenário. Essa ação é feita através do reconhecimento dos modelos apresentados como entrada por um sistema que faz a busca por modelos semelhantes em um banco de dados. Esse processo é chamado de recuperação de modelos, sendo que a correspondência e a indexação geralmente fazem parte desse processo. Embora a tecnologia tenha se desenvolvido, a interpretação robusta de cenas do ambiente ainda é um desafio para a computação em geral, dificultando áreas que utilizam da Visão Computacional, como por exemplo a Robótica. Levando em consideração a necessidade de novos algoritmos para a recuperação de formas, este trabalho objetiva aprofundar os estudos na área de Visão Computacional, propondo um novo algoritmo capaz de fazer a recuperação de formas tridimensionais em grandes bancos de dados. Embora os critérios de similaridade sejam importantes, a indexação das características para a busca de objetos semelhantes também tem a sua importância e impacta diretamente em velocidade e qualidade para o problema. Por este motivo, o foco deste trabalho é propor um novo método para a recuperação de modelos tridimensionais, utilizando a estrutura de dados chamada árvore de vocabulários, que apresenta resultados significativos na literatura para consultas por modelos bidimensionais. Neste trabalho, o método é apresentado de forma detalhada e é implementado através de um algoritmo. Os resultados obtidos por este algoritmo são apresentados e analisados utilizando bases de dados e métricas, os quais foram empregados em trabalhos anteriores. O método apresenta resultados promissores, porém com grande espaço para melhorias.

Palavras-chave: Visão Computacional. Recuperação de Modelo 3D. Modelos 3D. Árvore de Vocabulários. Escalabilidade.

ABSTRACT

Interpreting a scene from the computational perspective, usually is summarized by the interpretation and classification of its elements, realizing how they are organized by the scenario. This is done by recognizing the models presented as the input to a system that looks for similar models in a database. This process is named models retrieval, where matching and indexing are usually part of the process. Although technology has developed, the robust interpretation of scenes from the environment is still a challenge for computers in general, hindering areas that use Computer Vision, such as Robotics. Given the need for new shapes retrieval algorithms, this work aims to deepen the studies in the area of Computer Vision, proposing a new algorithm capable of recovering three-dimensional shapes in large databases. Although the similarity criteria are important, the indexing of characteristics for the search of similar objects also have their importance and directly impacts the speed and quality of the problem. For this reason, the focus of this work is to propose a new method for the retrieval of three-dimensional models, using the data structure called vocabulary tree, which presents significant results in the literature for queries by two-dimensional models. In this work, the method is presented in detail and is implemented through an algorithm. The results obtained by this algorithm are presented and analyzed using databases and metrics, which were used in previous works. The method presents promising results, but with large room for improvement.

Key-words: Computer Vision. 3D Model retrieval. 3D Models. Vocabulary Tree. Scalability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

2D bidimensional

3D tridimensional

4D quadridimensional

LISTA DE SIGLAS

CoSPAIR *Colored SPAIR*

dFPFH *Differential Fast Point Feature Histogram*

ORB *Oriented FAST and rotated BRIEF*

RGB-D *Red-Green-Blue-Depth*

SPAIR *Histograms of Spatial Concentric Surflet-Pairs*

Surflets Pontos 3D com coordenadas espaciais + vetor normal ao plano

LISTA DE SÍMBOLOS

α	Característica Alfa
β	Característica Beta
δ	Característica Delta
γ	Característica Gama

SUMÁRIO

1 INTRODUÇÃO

A necessidade por conhecer o mundo ao seu redor possibilitou ao homem construir desde as mais simples ferramentas, até os mais complexos conceitos que, ainda hoje, podem não ser compreendidos em sua totalidade. Um desses conceitos é a Matemática. Ela possibilitou à Ciência descrever com maior rigor os fenômenos naturais e, até hoje, serve de base para a formalização das mais diversas teorias, sendo a base mais fundamental da Ciência da Computação. A Matemática, resumidamente, está presente em todos os lugares.

Embora exista a ideia de que as operações de soma e subtração sejam as mais básicas realizadas por um ser humano, ela está equivocada. Antes mesmo de somar ou subtrair, é preciso necessariamente identificar os elementos presentes na operação. Esse pensamento leva a uma reflexão interessante: Como identificar esses elementos? Essa dúvida tem guiado pesquisadores de diversas áreas do conhecimento na busca por respostas, dando origem a áreas como a Inteligência Artificial e suas derivações, como por exemplo o Aprendizado de Máquina e a Visão Computacional. A Visão Computacional é uma das áreas que possui em aberto, em seu cerne, o problema de reconhecimento de padrões através do uso de métodos matemáticos e algorítmicos, buscando replicar as funcionalidades básicas da visão nos seres humanos.

A Visão Computacional está intimamente ligada ao sentido da visão, a qual procura reproduzi-lo utilizando recursos computacionais. Isso se deu por conta da visão ser o principal sentido responsável pela informação que grande parte dos seres vivos conseguem extrair do ambiente ao seu redor. Uma das características da visão é a facilidade com a qual a interpretação do meio é realizada por ela, sem nenhum grande esforço perceptível para este fim. Através desta interpretação é possível identificar os elementos ao redor dos seres vivos, e é somente após essa interpretação que decisões podem ser tomadas (DAVIES, 2012).

Interpretar uma cena, do ponto de vista computacional, geralmente se resume a interpretar e classificar suas partes, percebendo como elas estão organizadas pelo cenário. A interpretação das partes da cena é feita através do reconhecimento dos modelos apresentados como entrada por um sistema que faz a busca por modelos semelhantes em um banco de dados. Esse processo é chamado de recuperação de modelos. A correspondência e a indexação geralmente fazem parte desse processo. Correspondência é o processo de determinar a semelhança entre duas formas. Geralmente a semelhança é dada por uma medida de distância. Já a indexação é o processo de construção de uma estrutura de dados que serve para acelerar a busca por formas semelhantes (TANGELDER; VELTKAMP, 2004).

A interpretação robusta de cenas do ambiente ainda é um desafio para a computação em geral. Embora seja um desafio, as técnicas de reconhecimento e classificação estão cada vez mais avançadas, em grande parte pela evolução da tecnologia de *hardware*,

que não só proporcionou um aumento na capacidade de processamento dos computadores, como também na quantidade de modelos disponíveis em bancos de dados (*Google 3D Warehouse*[1], *Yobi3D*[2], *Yeggi*[3]) por consequência da intensa transmissão de informações através da *Internet*, o que possibilitou um aumento de recursos visuais para a Robótica.

Durante muito tempo as diversas formas de trabalhos eram feitas de maneira totalmente manual, porém, com a evolução da tecnologia, foi possível minimizar os custos e desenvolver novas técnicas, fazendo com que parte do trabalho seja realizado automaticamente por sistemas computadorizados. Para tarefas simples isso não é difícil, no entanto, quando a complexidade das tarefas é aumentada, a visão se torna um fator importante para que a eficiência desejada seja alcançada. Por este motivo, algoritmos eficientes para interpretação de cenas são uma necessidade atual, principalmente no campo da automação.

Levando em consideração a necessidade de novos algoritmos para a recuperação de formas (*i.e.*, modelos de objetos tridimensional (3D)), neste trabalho, são investigadas e aplicadas técnicas de extração e indexação de características (descritores) de objetos 3D para a confecção de um algoritmo robusto para esse fim. O trabalho se divide em duas partes importantes, sendo elas o treinamento e a consulta. A primeira consiste em extrair os descritores das formas e indexá-los em uma estrutura adequada (árvore de vocabulários), já a segunda tem como objetivo procurar modelos semelhantes a partir de uma entrada através da propagação dos seus descritores pela estrutura, calculando pontuações e ranqueando os modelos por semelhança.

Este capítulo apresenta, na seção 1.1, a motivação para o desenvolvimento do trabalho. Logo em seguida é exposta a metodologia utilizada, na seção 1.2. Na seção 1.3, estão os objetivos gerais e específicos deste trabalho. Por fim, na seção 1.4, é apresentada uma visão geral sobre os demais capítulos e como estão estruturados.

1.1 Motivação

A informação está espalhada em todos os cantos da *Internet*, seja ela por meio de textos, imagens, vídeos, ou até mesmo áudios. Porém, nem toda a informação disponível agrega conhecimentos úteis para seu receptor. A definição do que é conhecimento útil é pessoal e essa definição tende a mudar ao longo do tempo. São necessárias ferramentas de busca que consigam filtrar a informação de forma eficiente de acordo com a definição momentânea do que é útil ou não para um determinado indivíduo.

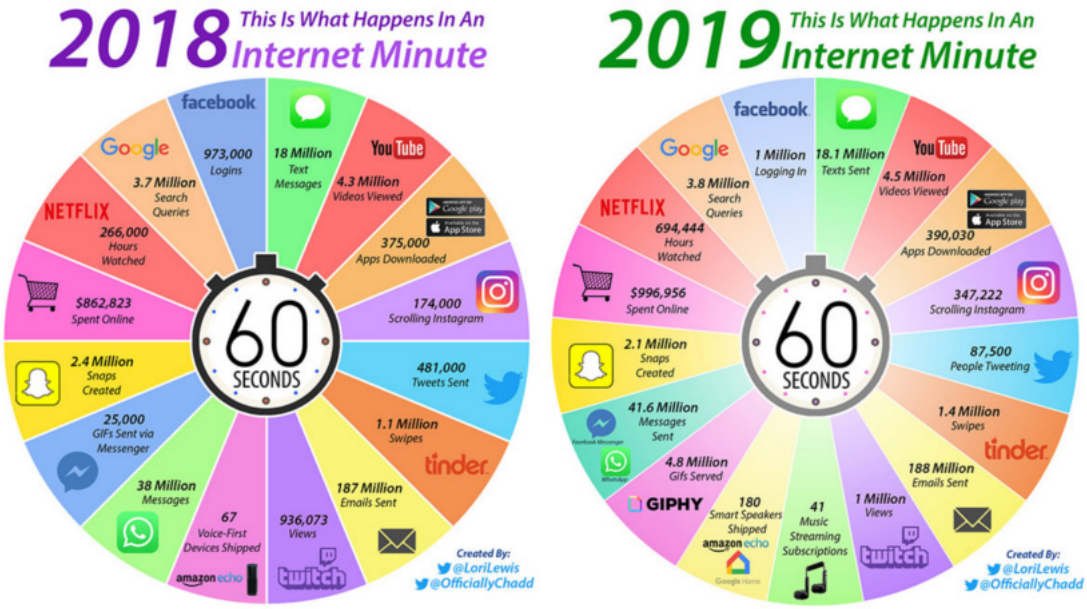
Nas últimas duas décadas houve um aumento considerável no tráfego da *Internet*, indo de 100 GB por dia para 46,600 GB por segundo, isso somente considerando entre os anos de 1992 e 2017 (CISCO, 2017). O aumento de dispositivos com conexão à *In-*

[1] Disponível em <https://3dwarehouse.sketchup.com/>
[2] Disponível em <http://yobi3d.com>
[3] Disponível em <https://www.yeggi.com/>

ternet vem crescendo mais rapidamente do que os próprios usuários de *Internet*, tendo a popularização dos *smartphones* contribuído muito para isso. A previsão é que até 2022 esses dispositivos representem cerca de 50% de todo o tráfego, superando os computadores pessoais e sua porcentagem de 49% em 2017, que deve cair para 19% em 2022 (CISCO, 2017).

Na Figura 1 é possível ver uma comparação feita por Desjardins (2019) sobre o que acontece na *Internet* em apenas um minuto nos anos de 2018 e 2019. Nesta imagem pode-se visualizar um grande aumento no consumo de conteúdo em vídeos e imagens representado principalmente pelos serviços de *streaming* (Netflix[4], YouTube[5], Twitch[6]) e redes sociais (Instagram[7]). Esse acréscimo de informação não vem somente deste último ano, mas sim de uma longa jornada de adaptação motivando a pesquisa por técnicas de interpretação e recuperação de imagens.

Figura 1 – O que acontece em um minuto na *Internet* em 2018 e 2019?



Fonte: (DESJARDINS, 2019).

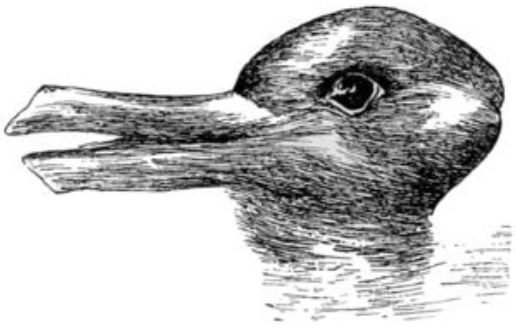
Geralmente, sistemas de busca são baseados em palavras-chave e realizam comparações entre nomes de arquivos, conteúdo ou contexto. Embora esse tipo de consulta seja simples de entender e implementar, ainda é relativamente ineficiente, já que requer muito recurso para catalogar cada modelo(ZOU; ZHANG, 2018). Além disso, nem todos os modelos podem ser bem definidos por palavras, dado que essas dependem de idioma,

[4] Disponível em <https://www.netflix.com/browse>
[5] Disponível em <https://www.youtube.com/>
[6] Disponível em <https://www.twitch.tv/>
[7] Disponível em <https://www.instagram.com/?hl=pt-br>

época, gênero e até mesmo de contexto (TANGELDER; VELTKAMP, 2004; IYER et al., 2005). Tendo isso em vista, uma abordagem baseada em conteúdo é a mais indicada para a consulta de modelos, visto que é simples para o usuário e não necessita de anotações.

Imagens e vídeos são as principais representações de informação utilizadas para algoritmos de busca e interpretação baseadas em conteúdo, principalmente por estarem intimamente ligadas com nossa visão. Apesar de serem boas representações, ainda apresentam o grande obstáculo da perspectiva por não contarem com as informações geométricas dos modelos. Isso pode facilmente ser observado na Figura 2. Nela, dois animais podem ser vistos dependendo do foco dado para a imagem pelo seu observador: um pato ou um coelho. Uma alternativa viável para contornar este problema é a utilização de objetos 3D como modelos de consulta. Nesse contexto, este trabalho é uma aposta no futuro, pois embora a procura por modelos 3D seja uma realidade, ainda não é tão popular e dificilmente é usada. Mesmo ainda não sendo muito utilizada, ou talvez até justamente por isso, a busca por esses modelos é uma área de pesquisa rica que deve ser explorada, principalmente por ter uma grande relação com problemas em Robótica no que diz respeito ao reconhecimento do ambiente em contexto e seus objetos.

Figura 2 – Ilusão visual do coelho-pato.



Fonte: (JASTROW, 1899).

Além do uso na Robótica e na busca por objetos semelhantes, a recuperação por modelos 3D pode ser utilizada na procura de casas, edifícios, pontos turísticos e até mesmo na identificação de pessoas através do reconhecimento de seus rostos, práticas comuns usando recuperação por imagens. Entretanto, com a popularização das impressoras 3D,

câmeras *Red-Green-Blue-Depth* (RGB-D) e tantas outras tecnologias, a busca utilizando objetos 3D apresenta aplicações únicas, como por exemplo a procura por projetos que utilizem determinado modelo como parte de sua impressão.

Este trabalho tem base no artigo dos autores Wahl, Hillenbrand e Hirzinger (2003) que propõe uma análise estatística com base em um recurso quadridimensional (4D). Os autores utilizam seis critérios para validar a recuperação de objetos 3D. Embora os critérios de validação sejam importantes, a indexação das características para a busca de objetos semelhantes também tem a sua importância e impactam diretamente em velocidade e qualidade para o problema (TANGELDER; VELTKAMP, 2004). Por esse motivo, este trabalho tem como objetivo a pesquisa por estruturas de indexação robustas para a recuperação de modelos 3D, em especial a árvore de vocabulário, que apresentou bons resultados na indexação de grandes volumes de imagens (NISTER; STEWENIUS, 2006; PAZ, 2018).

1.2 Metodologia

A seguir são expostos os principais passos da metodologia utilizada para obtenção dos objetivos do trabalho.

1. **Definir um tema para o trabalho:** Escolher um tema na área de inteligência artificial que seja interessante e desafiador e, a partir disso definir uma linha de pesquisa;

2. **Explorar o estado da arte de acordo com o tema proposto:** Fazer um estudo dos trabalhos obtidos através de métricas de busca;

3. **Selecionar os trabalhos relacionados:** Filtrar através do senso crítico os trabalhos que realmente são relevantes e não repetitivos para o objetivo proposto;

4. **Desenvolver protótipo:** Programar e aplicar o que foi aprendido no desenvolvimento do protótipo baseado nas técnicas encontradas na literatura;

5. **Obter base de dados:** Investigar objetos disponíveis na rede, para montar uma base de objetos robusta e diversificada;

6. **Desenvolver o algoritmo proposto:** Dar continuidade ao protótipo, incrementando o algoritmo com árvore de vocabulários;

7. **Realizar os testes e registrar os resultados:** Realizar os testes e avaliar os resultados a partir de métricas. Registrar os resultados e avaliações;

8. **Registrar trabalho realizado:** Registrar os conceitos, técnicas utilizadas e formas de validação do algoritmo proposto no trabalho de conclusão de curso.

1.3 Objetivos

Este trabalho tem como objetivo a elaboração de um algoritmo de recuperação de objetos 3D, que seja escalável para grandes quantidades de modelos em diferentes condições. O algoritmo deve trabalhar de maneira que receba como entrada um arquivo contendo um objeto de exemplo e retorne os objetos semelhantes presentes na base de dados.

São definidos como objetivos específicos os tópicos a seguir:

- Implementar um algoritmo existente na literatura para compreendê-lo e realizar um estudo comparativo com o algoritmo proposto neste trabalho;
- Obter e criar bases de objetos 3D variados para validação do algoritmo da literatura e do algoritmo proposto;
- Validar o algoritmo da literatura e o algoritmo proposto para as bases de objetos 3D.

1.4 Organização do Documento

1. Capítulo 2 - Fundamentação Teórica: uma breve apresentação dos conceitos principais para o entendimento do trabalho;
2. Capítulo 3 - Trabalhos Relacionados: apresenta os critérios de busca utilizados e os trabalhos relevantes filtrados por elas;
3. Capítulo 4 - Desenvolvimento: apresenta o caminho seguido para alcançar os resultados apresentados;
4. Capítulo 5 - Resultados: apresenta os resultados alcançados até o momento;
5. Capítulo 6 - Considerações Finais: apresenta as considerações finais da pesquisa realizada, apontando os principais tópicos observados.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados brevemente os principais conceitos para o entendimento deste trabalho, começando pelo conceito de Visão Computacional, apresentado na seção 2.1. Uma breve introdução é realizada, diferenciando modelos bidimensional (2D) de modelos 3D na seção 2.2. Logo em seguida, é apresentada a extração de características na seção 2.3. Por fim, os conceitos de *clustering* e indexação de características são apresentados nas seções 2.4 e 2.5, respectivamente.

2.1 Visão Computacional

A Visão Computacional é a sub-área da Ciência da Computação que estuda como percebemos o meio em que vivemos e tenta replicar essa percepção em computadores. Nos últimos anos tem ganhado destaque, evoluindo suas técnicas, principalmente de recuperação de modelos. Mesmo com os avanços dos últimos anos, tanto em *hardware* quanto em *software*, ainda não é possível fazer um computador interpretar uma cena com a riqueza e profundidade que um humano interpreta. Essa dificuldade se dá por simplesmente não conseguirmos descrever de forma eficiente o ambiente em que vivemos e nem mesmo como vemos esse ambiente (SZELISKI, 2010).

Embora seja muito associada com a visão humana, a Visão Computacional estuda a percepção do meio utilizando quaisquer fontes de dados, como sons e polígonos. Consequentemente, existem diversas formas para tratar os inúmeros tipos de fontes de dados que podem ser capturados com diferentes tipos de sensores.

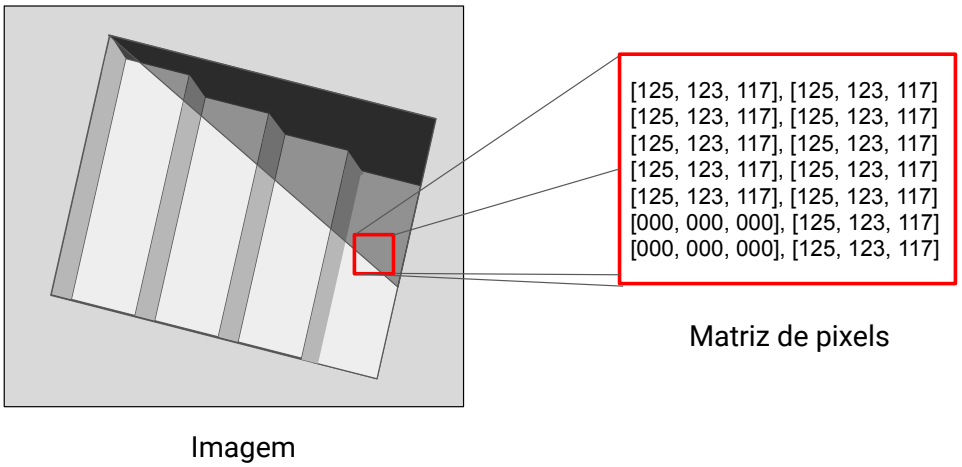
A recuperação de modelos é um dos principais campos de pesquisa da Visão Computacional. Recuperar um modelo significa, do ponto de vista computacional, obter uma lista com os modelos semelhantes ao modelo de consulta ordenados pela sua similaridade. A correspondência e a indexação geralmente fazem parte desse processo. Um dos seus principais desafios ainda hoje é construção de métodos escaláveis para grandes bases de dados.

2.2 Modelos 2D e Modelos 3D

Imagens ou modelos 2D (representados na Figura 3), para um computador, nada mais são do que matrizes com informações sobre os *pixels* distribuídos ao longo dos seus campos. Cada *pixel* de uma imagem pode assumir diversos valores para representar as cores em uma imagem, geralmente em combinações de verde, vermelho e azul. Já os modelos 3D, ilustrados na Figura 4, são representações de objetos através de informações espaciais obtidas através de coordenadas no espaço euclidiano.

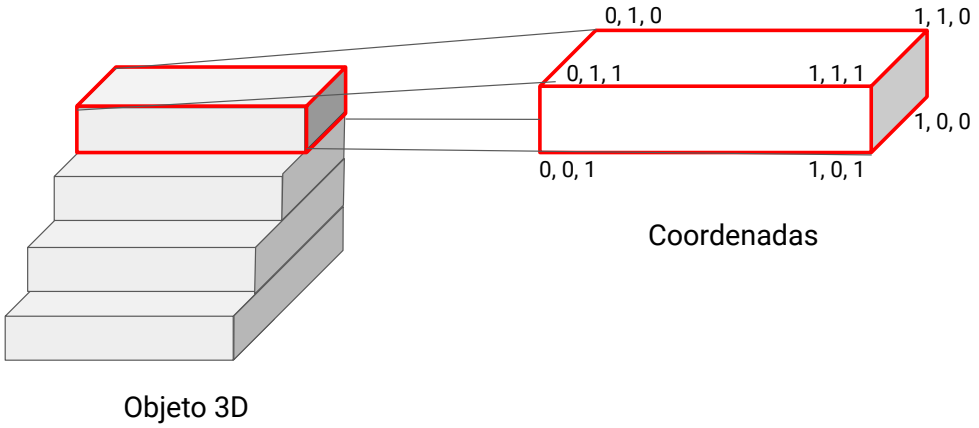
Existem diferentes formas para representar um modelo 3D em arquivos de texto, cada uma com suas peculiaridades feitas para problemas específicos. Uma forma popular são as malhas poligonais que, apesar de simples, conseguem proporcionar uma visão

Figura 3 – Representação de uma imagem em um computador.



Fonte: Elaborado pelo autor.

Figura 4 – Representação de um objeto em um computador.



Fonte: Elaborado pelo autor.

clara do objeto. O formato consiste em descrever cada vértice e face do objeto através de coordenadas tridimensionais globais. Assim, um vértice consiste em ponto no espaço,

diferente de uma face, que consiste em um conjunto de pontos. As faces geralmente são triangulares por conveniência. Este formato também apresenta suas desvantagens, sendo a principal delas a necessidade de uma grande quantidade de memória. Por usar um sistema de coordenadas globais, em algumas situações é preciso fazer transformações geométricas em suas coordenadas para alinhar o objeto de interesse ao quadro do modelo do objeto referenciado antes que o correspondência possa ser realizada. É necessário um esforço para que ajustes excessivos não sejam feitos e isto reforça a busca por uma representação compacta, robusta, que não dependa de um quadro de coordenadas global e que tenha a capacidade descritiva de distinguir formas arbitrárias (WAHL; HILLENBRAND; HIRZINGER, 2003).

2.3 Extração de Características

Modelos 2D e 3D não apresentam informações relevantes para aplicações computacionais sem passar por um processo de extração de características. Isso acontece porque computadores não conseguem processar informação como humanos, sendo o seu processo de reconhecimento de padrões puramente matemático. Para solucionar este problema, existe uma série de algoritmos de extração de características responsáveis por dar sentido para imagens e objetos analisados por computadores.

2.3.1 Relação Par-*Surflets*

Um dos algoritmos de extração de características de modelos 3D amplamente utilizado foi proposto pelos autores Wahl, Hillenbrand e Hirzinger (2003). Neste método, as características do modelo são calculadas através de equações de geometria analítica e álgebra linear para descrever a relação entre seus polígonos. O resultado é um conjunto de vetores de características 4D que são definidas pela interação entre todos os pares de *surflets* - pontos 3D (p) com coordenadas espaciais + vetor normal ao plano ($\mathbf{n}$). As relações entre os pares de *surflets* podem ser vistas como uma generalização de curvaturas e são extraídas de polígonos com no mínimo três pontos distintos (triângulos). As equações necessárias para calcular essas características podem ser vistas a seguir e utilizam os símbolos: $(\cdot)$ denota o produto escalar entre dois vetores, $(\times)$ denota o produto vetorial entre dois vetores, $\|\cdot\|$ denota a norma de um vetor, e $|\cdot|$ denota o módulo de um número real. Se a Equação 2.1 for satisfeita, o ponto p_1 é escolhido como origem, caso contrário, p_2 é a origem

$$| \mathbf{n}_1 \cdot (p_2 - p_1)| \leq | \mathbf{n}_2 \cdot (p_2 - p_1)|. \tag{2.1}$$

Se p_2 for escolhido como origem, as posições de p_1 e p_2 nas equações a seguir devem ser trocadas. Assumindo que p_1 é a origem, os vetores base do sistema de coordenadas são então definidos pelas equações 2.2, 2.3 e 2.4

$$\mathbf{u} = \mathbf{n}_1, \tag{2.2}$$

$$\mathbf{v} = \frac{(p_2 - p_1) \times \mathbf{u}}{\|(p_2 - p_1) \times \mathbf{u}\|}, \tag{2.3}$$

$$\mathbf{w} = \mathbf{u} \times \mathbf{v}. \tag{2.4}$$

As relações entre os pares de *surflets* (p_1, $\mathbf{n}_1$) e (p_2, $\mathbf{n}_2$) são descritas pelos parâmetros das equações 2.5, 2.6, 2.7 e 2.8, e definem o vetor de características $S = (\alpha, \beta, \gamma, \delta)$ detalhado na Figura 5, sendo o $arctan(x, y)$ dado pela Equação 2.9. Os atributos definidos como α e β representam a normal $\mathbf{n}_2$ como um ângulo azimutal e o cosseno de um ângulo polar, respectivamente. Os outros dois, γ e δ, representam a direção e o comprimento da translação de p_1 para p_2, respectivamente

$$\alpha = arctan(\mathbf{w} \cdot \mathbf{n}_2, \mathbf{u} \cdot \mathbf{n}_2), \tag{2.5}$$

$$\beta = \mathbf{v} \cdot \mathbf{n}_2, \tag{2.6}$$

$$\gamma = \mathbf{u} \cdot \frac{p_2 - p_1}{\|p_2 - p_1\|}, \tag{2.7}$$

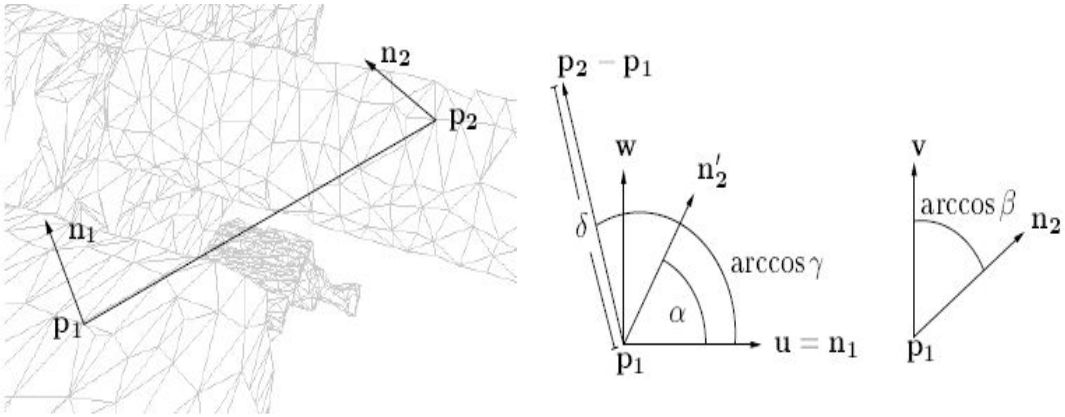
$$\delta = \|p_2 - p_1\|, \tag{2.8}$$

$$arctan(x, y) = \begin{cases} arctan(y/x) & se\ x > 0 \wedge y > 0, \\ arctan(y/x) + \pi & se\ x < 0, \\ arctan(y/x) + 2\pi & se\ x > 0 \wedge y < 0. \end{cases} \tag{2.9}$$

2.4 *Clustering*

Clustering é um processo de classificação não supervisionada, que basicamente agrupa dados de um conjunto de entrada com base em sua similaridade. *K-Medians* (ANDERSON et al., 2006), *K-Medoids* (CAO; YANG, 2010), *K-Means* (JAIN, 2010) e *K-Majority* (GRANA et al., 2013) são alguns exemplos de algoritmos de *clustering*. Esses algoritmos geralmente apresentam uma primeira fase comum, escolhendo aleatoriamente valores centrais que representam um grupo de dados (isto é, um *cluster*), de modo que tais valores podem pertencer ao conjunto ou não. As principais diferenças entre os algoritmos de *clustering* são: como o algoritmo atualiza o valor central de cada *cluster* e quais tipos de dados de entrada são suportados pelo algoritmo.

Figura 5 – Relação entre pares de *surflets*



Fonte: (WAHL; HILLENBRAND; HIRZINGER, 2003).

2.4.1 *K-Means*

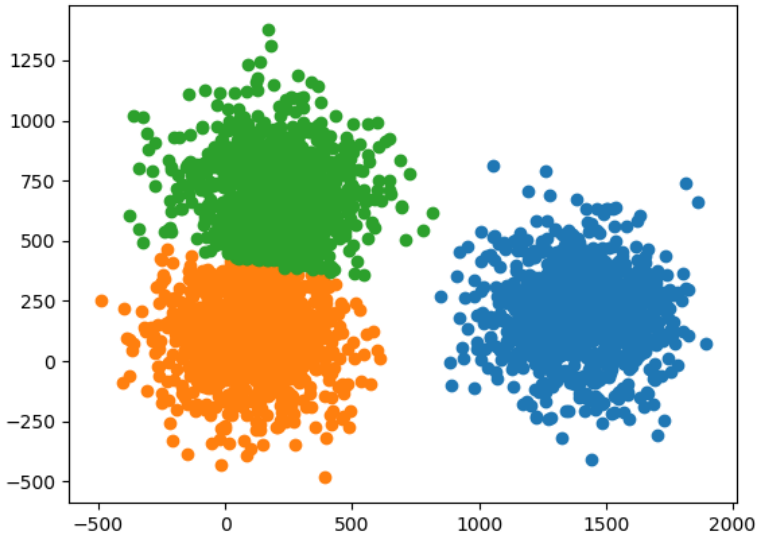
O método de *K-Means*, inicialmente proposto por MacQueen et al. (1967), é um dos algoritmos mais populares de classificação não supervisionada. Seus dados de entrada são compostos por coordenadas do espaço euclidiano. Esse algoritmo começa escolhendo valores de coordenadas aleatórios para representar k centróides, sendo que cada um dos centróides representam o valor central de um grupo de dados. Após os centróides serem iniciados, cada dado d é associado com um centróide mais próximo. O algoritmo continua recalculando os k centróides com base nos dados e associando os dados aos novos centróides. Esse processo continua durante t iterações ou até que nenhum dado d tenha mudado seu centróide.

Um resultado do algoritmo utilizando k igual a 3 pode ser observado na Figura 6. Na figura é possível observar três grupos de dados, representados por cores distintas. Cada grupo de dados possui um ponto central, chamado centróide, o qual representa o conjunto de dados do *cluster*.

2.5 Indexação de Características

A indexação das características é o foco principal deste trabalho e é realizada através da construção de uma estrutura de dados que serve para acelerar a busca dos modelos. Para que um algoritmo de recuperação de modelos seja eficiente, tanto a extração

Figura 6 – Resultado da execução do algoritmo de *K-Means* para três *clusters*. Cada círculo representa um dado, e as cores codificam os clusters.



Fonte: Elaborado pelo autor.

das características, quanto a sua indexação devem ser eficientes.

2.5.1 Histograma de Relação Par-*Surflets*

Após a fase de extração de características, Wahl, Hillenbrand e Hirzinger (2003) apresentam uma indexação simples, apenas mapeando os vetores de características que representam o modelo em um histograma para posterior comparação. A construção dos histogramas para cada modelo é feita a partir de um mapeamento $h(x)$ de um vetor de características s em uma caixa i do histograma, sendo $s \in S$, o qual é o conjunto de todos os vetores de características do modelo, e $i \in I$, que é o conjunto de d caixas. Para isso, os dados não normalizados entre 0 e 1, de forma que a soma de todas as caixas do histograma seja 1. Assim, o histograma H é obtido pela Equação 2.10

$$H(i) = \frac{card\{s \in S \mid h(s) = i\}}{cardS}. \tag{2.10}$$

A partir disso, alguma medida de distância pode ser usada para calcular a dissimilaridade dos histogramas gerados.

2.5.2 Árvore de Vocabulários

Inspirados por Sivic e Zisserman (2003), Nister e Stewenius (2006) propuseram um método de indexação escalável que quantifica hierarquicamente os descritores em

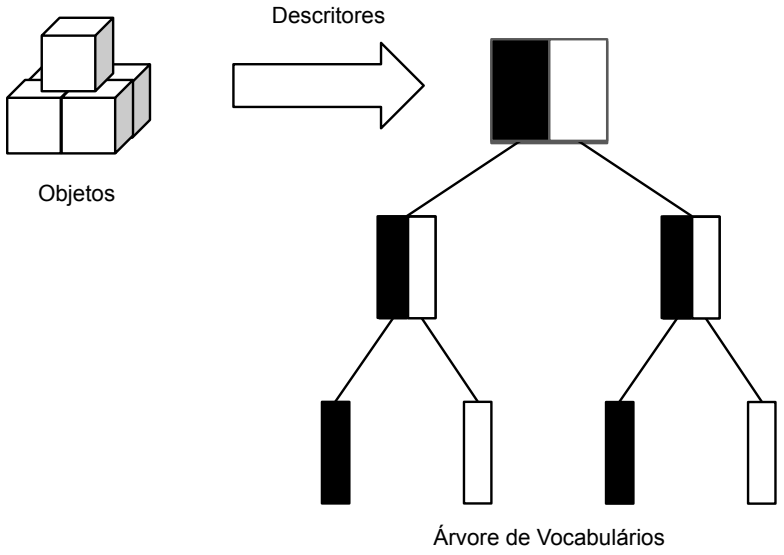
uma árvore de vocabulários, permitindo que um vocabulário consideravelmente extenso e discriminatório seja usado com eficiência. O autores também mostraram que uma melhor qualidade de recuperação é obtida quando uma base de dados maior é utilizada.

2.5.2.1 Construção da Árvore de Vocabulários

No início da fase de treinamento do algoritmo, os descritores são extraídos de todos os modelos de uma base de dados e adicionados em um lista de descritores. Essa lista é composta por todos os descritores de todos os modelos de treinamento. A lista é então passada como entrada para o nodo raiz da árvore de vocabulários para que seja propagada e dê início aos próximos níveis da árvore.

As subárvores são criadas a partir da *clusterização* da lista de descritores do nodo pai, gerando k listas que devem ser passadas aos seus k filhos, como ilustrado na Figura 7. Nela, os cubos representam os objetos da base de dados, que tem seus descritores extraídos e passados para serem *clusterizados* no nodo raiz da árvore (representados pela seta e pelo quadrado, respectivamente). Cada *cluster* da árvore é representado por um retângulo na cor preta ou branca. Este processo é aplicado recursivamente para cada subárvore, construindo a árvore de vocabulários. O processo chega ao fim quando um nível l é alcançado ou quando o número de características da lista da subárvore for menor ou igual a k. Resumidamente, o parâmetro k define a quantidade de filhos e o número de *clusters* utilizados, enquanto que o parâmetro l define o nível máximo da árvore.

Figura 7 – Construção da árvore de vocabulários para $k = 2$ e $l = 2$.



Fonte: Elaborado pelo autor.

2.5.2.2 Sistema de Pontuação e Consulta com a Árvore de Vocabulários

Durante a construção da árvore de vocabulários, são calculados os pesos de cada nodo e modelo. Para cada nodo i da árvore é dado um peso w_i, definido pela Equação 2.11, sendo N o número total de modelos e N_i o número de modelos da base de dados que teve pelo menos um descritor passando pelo nodo i, onde $i \in I$, o qual é o conjunto com os índices de todos os nodos da árvore. Quanto mais modelos têm seus descritores passando por um nodo, menos discriminante este nodo é.

$$w_i = \ln \frac{N}{N_i} \tag{2.11}$$

Para calcular o peso dos modelos, para cada modelo $j \in J$, onde J é o conjunto com os índices dos modelos da base de dados, a Equação 2.12 é utilizada, na qual d_{ij} é o peso do modelo, m_i é a quantidade de descritores do modelo j que passaram pelo nodo i e w_i é o peso do nodo i. Se nenhum descritor do modelo j passar pelo nodo i, d_{ij} será zero.

$$d_{ij} = m_{ij} w_i \tag{2.12}$$

Durante a fase de busca, um modelo é recebido como consulta, suas características são extraídas e propagadas pela estrutura de indexação, comparando-as com os centroides de cada nível da árvore. Enquanto a propagação acontece, são calculados seus pesos q_i, dados pela Equação 2.13, para cada nodo i. n_i é a quantidade de descritores do modelo de consulta que passaram pelo nodo i e w_i é o peso do nodo i.

$$q_i = n_i w_i \tag{2.13}$$

Ao fim da descida das características do modelo de consulta, seus q_i são compilados em um vetor q, com $q = (q_{i_1}, q_{i_2}, \ldots, q_{i_n})$, onde $\{i_1, i_2, \ldots, i_n\} = I$. Para cada modelo da base de dados, seus d_i são recuperados e compilados em um vetor d_j, com $d_j = (d_{i_1 j}, d_{i_2 j}, \ldots, d_{i_n j})$, onde $(i_1, i_2, \ldots, i_n) = I$. Por fim, a dissimilaridade é dada por um pontuação obtida pela Equação 2.14. Quanto menor o valor obtido, maior é a semelhança entre os modelos avaliados.

$$s(q, d_j) = \left\| \frac{q}{\|q\|} - \frac{d_j}{\|d_j\|} \right\| \tag{2.14}$$

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados de forma resumida os trabalhos que, de alguma forma, contribuíram com a presente monografia. Com este propósito a seção 3.1 a seguir mostra, passo a passo, a jornada realizada para fazer o levantamento dos trabalhos relacionados e seus critérios de seleção. Já na seção 3.2, são apresentados os trabalhos de fato, onde uma breve explicação sobre eles é dada, apresentando seus pontos importantes. Por fim são apresentadas as considerações do capítulo, revelando a importância da pesquisa realizada.

3.1 Método de Pesquisa

Este trabalho é fortemente inspirado nos trabalhos de Paz (2018) e Silva (2018). Destes dois, o artigo que serviu de base para cada um foi extraído e adicionado aos trabalhos relacionados desta monografia. Os demais trabalhos foram obtidos a partir de *strings* de busca que serviram para filtrar e direcionar a pesquisa.

Antes de realizar a pesquisa de fato, um escopo foi definido contendo os objetivos por trás das *strings* de busca. Definir um objetivo serve principalmente para a busca não se perder ou ficar sem sentido. O escopo foi definido para descobrir quais são as técnicas recentes de reconhecimento de objetos 3D, além de buscar artigos que apresentam uso da árvore de vocabulários como forma de indexação de descritores para reconhecimento de objetos 3D. Duas perguntas foram elaboradas, dando origem a duas *strings* que foram usadas na base de dados Scopus[1].

- Perguntas:
  1. Quais são os métodos de reconhecimento para modelos 3D?
  2. Existem métodos de reconhecimento de objetos 3D que utilizam árvores de vocabulário?
- *Strings* de busca:
  1. *TITLE-ABS-KEY (("3d object"OR "3d model") AND ("search"OR "recognition"OR "retrieval"OR "mat- ching"))*
  2. *TITLE-ABS-KEY (("3d object"OR "3d model") AND ("search"OR "recognition"OR "retrieval"OR "mat- ching") AND "vocabulary tree")*

Após ter os resultados das *strings* de busca, uma filtragem em duas partes foi realizada para eliminar falsos-positivos e extrair artigos que realmente tenham informações relevantes. A primeira obedecendo os critérios de escala de prioridade apresentados por Silva (2018). Já a segunda parte leva em conta os critérios de exclusão expostos por Paz

[1] Disponível em <https://www.scopus.com/home.uri>

(2018) com pequenas alterações, basicamente trocando a palavra "imagens"por "modelos 3D". Os títulos, resumos e conclusões dos artigos foram lidos, porém somente aqueles com alto grau de relevância tiveram uma análise mais aprofundada.

3.2 Trabalhos Selecionados

Inicialmente, a primeira *string* de busca retornou cerca de 8354 resultados - um número tão grande já era esperado, pois a primeira *string* de busca é bem genérica, buscando somente ter uma noção das técnicas utilizadas -, dos quais as 20 primeiras páginas de resultado foram lidas e filtradas através dos critérios discutidos na seção anterior. Destas 20 páginas, os três artigos que apresentaram contribuições relevantes para esta monografia (sua presença nesta seção pode ser facilmente defendida) foram adicionados nesta seção.

A segunda *string* teve um número bem menor de resultados, com apenas cinco artigos retornados, sendo que somente um deles se mostrou relevante. Como o artigo encontrado apresenta um método que tem como entrada uma seleção de imagens com diferentes perspectivas, pode-se concluir que ainda não existem - pelo menos não existem muitos - métodos que utilizam a árvore de vocabulário em conjunto com algoritmos de recuperação de objetos com a entrada por exemplo de modelos 3D, como o proposto nesta monografia.

No trabalho realizado por Wahl, Hillenbrand e Hirzinger (2003), um recurso 4D foi proposto para parametrizar a relação geométrica intrínseca de pares de Pontos 3D com coordenadas espaciais + vetor normal ao plano (*Surflets*), sendo seu conjunto a representação dos recursos globais e locais da superfície. As características são mapeadas em histogramas que, de forma geral, representam cada objeto. Também são comparados seis critérios de validação da recuperação de modelos 3D, sendo o critério de *Likelihood* o que apresenta melhores resultados para os testes realizados. Os autores afirmam que testes foram realizados com presença de ruídos e oclusões, demonstrando que o algoritmo ainda obtém taxas de reconhecimento maiores que 80% mesmo nesses casos.

Em Nister e Stewenius (2006), uma estrutura hierárquica chamada árvore de vocabulários foi proposta. A estrutura apresenta um sistema de pontuação para eleger as imagens mais semelhantes a uma dada consulta e escala eficientemente para um grande número de imagens. A técnica proposta é naturalmente robusta contra oclusão e confusão de fundo e sua qualidade é avaliada através de uma banco de dados com grupos de imagens conhecidos do mesmo objeto ou localização sob diferentes pontos de vista, variando inclusive a iluminação. Os autores afirmam que quanto maior o vocabulário usado, uma melhor qualidade de recuperação é obtida.

O trabalho de Silva (2018) foi inspirado em Wahl, Hillenbrand e Hirzinger (2003) e teve como objetivo propor um novo algoritmo de extração de descritores para a recuperação de objetos 3D. Além dos seis critérios apresentados no trabalho base, um novo

critério elaborado pelo autor é apresentado, chamado de *Variation Rate* e apresenta uma alternativa diferente dos critérios anteriores. Enquanto os critérios analisados por Wahl, Hillenbrand e Hirzinger (2003) comparam o histograma de um objeto de consulta com os histogramas dos objetos treinados pelo algoritmo, o novo critério busca comparar os histogramas já treinados com outros histogramas já treinados, recompensando os objetos semelhantes e penalizando os objetos diferentes. Diferentemente do trabalho base, o autor afirma que, em seus resultados, o critério de similaridade que apresentou melhores percentagens de acerto foi o critério *Kullback-Leibler*. O trabalho também apresentou a elaboração de uma nova sub-característica chamada Relação Esfera-*Surflets*, que analisa a distância entre cada *Surflets* até uma esfera que envolve o objeto.

Paz (2018) teve como objetivo a construção de um motor de busca por imagem que seja escalável para uma grande quantidade de modelos e robusto para as diferentes condições em que uma imagem pode se encontrar, utilizando apenas métodos livres. Para este fim, o algoritmo utilizado para a extração de características foi o *Oriented FAST and rotated BRIEF* (ORB) (RUBLEE et al., 2011) e o método utilizado para a classificação dos descritores foi o *K-Majority* (GRANA et al., 2013). O trabalho é fortemente baseado em Nister e Stewenius (2006) e também utiliza a árvore de vocabulários para a indexação das características extraídas das imagens. O algoritmo proposto utilizando somente código livre é capaz de superar alguns concorrentes patenteados.

Mars et al. (2015) utiliza técnicas de reconhecimento de objetos 3D baseadas em *hierarchically structured multi-view features* e em árvore de vocabulários para a indexação das características. O algoritmo é testado com um banco de dados com imagens de prédios tiradas de diferentes perspectivas. Para construir a árvore com as características foi aplicado recursivamente o método de *clustering, k-means* (JAIN, 2010). O autor também testou algoritmos baseados em reconhecimento de modelos 2D e comparou com os algoritmos baseados em modelos 3D usados. Os resultados dos testes mostraram que algoritmos de reconhecimento de modelos 3D superam os de modelos 2D por contar com mais importações descritivas dos modelos. É importante ressaltar que, neste artigo, a entrada dos dados foi definida como sendo as imagens de diferentes perspectivas de um objeto, porém nada impede dessas imagens serem geradas por algum algoritmo externo que converta um modelo 3D em múltiplas imagens, como é feito em algumas técnicas baseadas em *multi-view* para reconhecimento de objetos 3D.

Savelonas, Pratikakis e Sfikas (2016) introduzem um método de recuperação de objetos 3D parcial de forma híbrida, aplicável em nuvens de pontos e modelos 3D estruturados, que utiliza o método de codificação de *Fisher* (PERRONNIN; DANCE, 2007) em conjunto com o método *Differential Fast Point Feature Histogram* (dFPFH). A similaridade global da forma é estimada por meio de uma distância ponderada de vetores *Fisher* enquanto a avaliação de similaridade de forma local é dada pela média das distâncias mínimas ponderadas associadas com pares de valores dFPFH calculados na consulta parcial

e no objeto alvo respectivamente. O método proposto foi testado em duas bases de dados, sendo a primeira composto por 360 formas, separadas em 20 classes, e a segunda por 384 formas, divididas em 6 classes. Os resultados mostram que a recuperação de objetos 3D parciais requer a utilização de características locais e globais para um melhor resultado. O algoritmo também foi testado em um cenário real e, embora tenha resultados positivos para os dois bancos de dados anteriores, em uma situação real não apresenta um resultado significativo.

Mian, Bennamoun e Owens (2010) afirmam que a recuperação de objetos por recursos locais é eficiente, porém, se testado por força bruta em cada ponto do objeto, se torna uma aplicação muito custosa. Tendo isso em vista, os autores propõem um algoritmo de detecção de pontos-chave de acordo com a sua qualidade, selecionando somente os melhores pontos para extrair características locais invariantes de escala. Para um ponto do objeto ser considerado um ponto-chave ele deve atender a três requisitos, sendo eles: os pontos devem se repetir com frequência em diferentes modelos do mesmo objeto; uma única base de coordenadas 3D pode ser definida a partir da superfície de vizinhança para extrair características locais invariantes; a superfície de vizinhança do ponto chave deve conter informações descritivas suficientes para caracterizar unicamente esse ponto. Caso um grande número de pontos passem nos três requisitos, uma seleção desses pontos é feita. Os resultados mostram que a repetibilidade de um ponto tem relação direta com a qualidade dele.

Logoglu, Kalkan e Temizel (2016) propõe dois descritores 3D locais: *Histograms of Spatial Concentric Surflet-Pairs* (SPAIR) e *Colored SPAIR* (CoSPAIR). O primeiro com informações somente da forma e o segundo com informações de cores também. Os descritores utilizam relações entre pares de *Surflets* para extrair as características das formas criando, a partir dessas informações, histogramas que representam essas características. Os autores também dividem classificação em dois tipos: nível de categoria, quando os modelos são divididos previamente por rótulos de classe, e nível de instância, quando os modelos são divididos pelas informações da forma geométrica. A principal diferença entre as técnicas propostas se dá ao fato de que no método CoSPAIR as informações de cores também são utilizadas em histogramas de cores. Os resultados mostram que a informação das cores é extremamente relevante para o método de busca, apresentando melhores resultados que a técnica sem essas informações.

3.3 Considerações do Capítulo

Através da pesquisa realizada foi possível conhecer mais a respeito do processo de recuperação de modelos, tanto de imagens, quanto de objetos 3D. Embora este trabalho tenha sido pensado para objetos, entender os processos de extração e indexação de características para imagens proporcionou aprender os conceitos envolvidos na construção da árvore de vocabulários, já que não foi possível encontrar na literatura trabalhos que

utilizam esta estrutura para objetos de exemplo.

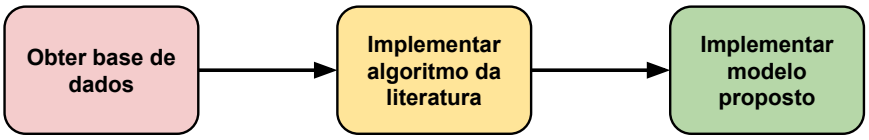
Também é possível observar o quanto um algoritmo de recuperação de modelos pode ser modificado e melhorado, tendo em vista a existência de diversos métodos para seleção de pontos de interesse, extração de descritores e indexação deles em estruturas. Até mesmo os dados utilizados como entrada podem ser modificados para conter informações de cores, melhorando o desempenho do algoritmo.

Durante o processo de busca, os trabalhos encontrados que utilizam a árvore de vocabulários para recuperação de objetos 3D não utilizam a consulta por modelo 3D de exemplo. Ao invés disso, utilizam diversas imagens, obtidas de diversos ângulos para construir a estrutura. Embora seja um método válido, a proposta deste trabalho é propor a indexação a partir de características obtidas por algoritmos de extração de descritores 3D, que contam com a informação da profundidade dos objetos, evitando falsos positivos causados por ilusões de perspectiva.

4 DESENVOLVIMENTO

Neste capítulo é apresentado o passo a passo realizado para o desenvolvimento deste trabalho, que consiste nas três principais fases mostradas na Figura 8. A primeira fase consiste em obter uma base de dados para realizar os testes envolvendo os algoritmos implementados e discutidos neste capítulo. A fase seguinte se dá através do desenvolvimento de um algoritmo baseado em um método encontrado na literatura. E por fim, a última etapa do desenvolvimento é a implementação do método proposto. Este capítulo está organizado da seguinte maneira: na seção 4.1 é apresentada a base utilizada, na seção 4.2 é apresentado o algoritmo de Wahl, Hillenbrand e Hirzinger e na seção 4.3, o método proposto.

Figura 8 – Fases do desenvolvimento deste trabalho.



Fonte: Elaborado pelo autor.

4.1 Base de Objetos

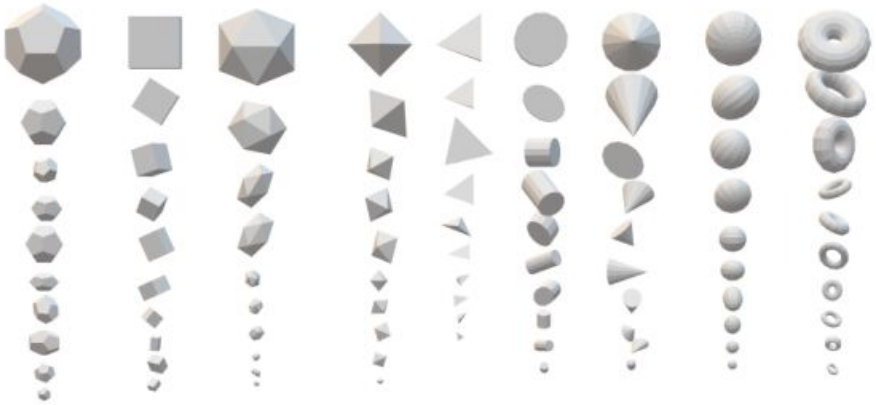
Para que seja possível validar os algoritmos implementados, primeiramente é necessário obter uma base de objetos 3D. Para isto, foram encontradas bases de objetos utilizadas por trabalhos anteriores realizados pela comunidade de Visão Computacional. A base de objetos 3D escolhida foi retirada do trabalho proposto por Silva (2018). Nesse trabalho, são apresentadas quatro bases de dados com objetos extraídos de *Princeton Shape Benchmark*[1] ou criados pelo próprio autor. Destas, as base 3 foi escolhida para este trabalho.

[1] Disponível em <https://shape.cs.princeton.edu/benchmark/>

Comumente, algoritmos de recuperação de modelos são avaliados com uma base de dados dividida em duas partes: base de treinamento e base de testes. Entretanto, para agilizar o processo de busca e testar se a aplicação consegue encontrar o próprio modelo de referência na base de dados, a mesma base de dados foi utilizada, tanto para a fase de treinamento, quanto para a fase de busca.

A base de objetos 3 foi criada para suprir a necessidade de validar os algoritmos com objetos simples e testar aspectos como invariância à escala, rotação e translação. Assim, foram criados os 90 objetos mostrados na Figura 9, cada um pertencente a uma de nove classes contendo dez objetos. As classes são divididas em: dodecaedro, hexaedro, icosaedro, octaedro, tetraedro, cilindro, cone, esfera e toróide. Para cada classe uma pasta foi criada contendo os 10 objetos, nominados de 0 à 9 seguindo o aumento da escala do número de polígonos, posição e rotação. Vale ressaltar que para dois objetos serem considerados semelhantes nesta base de dados, os mesmos devem pertencer à mesma classe de objetos.

Figura 9 – Base de objetos 3 - 90 objetos.



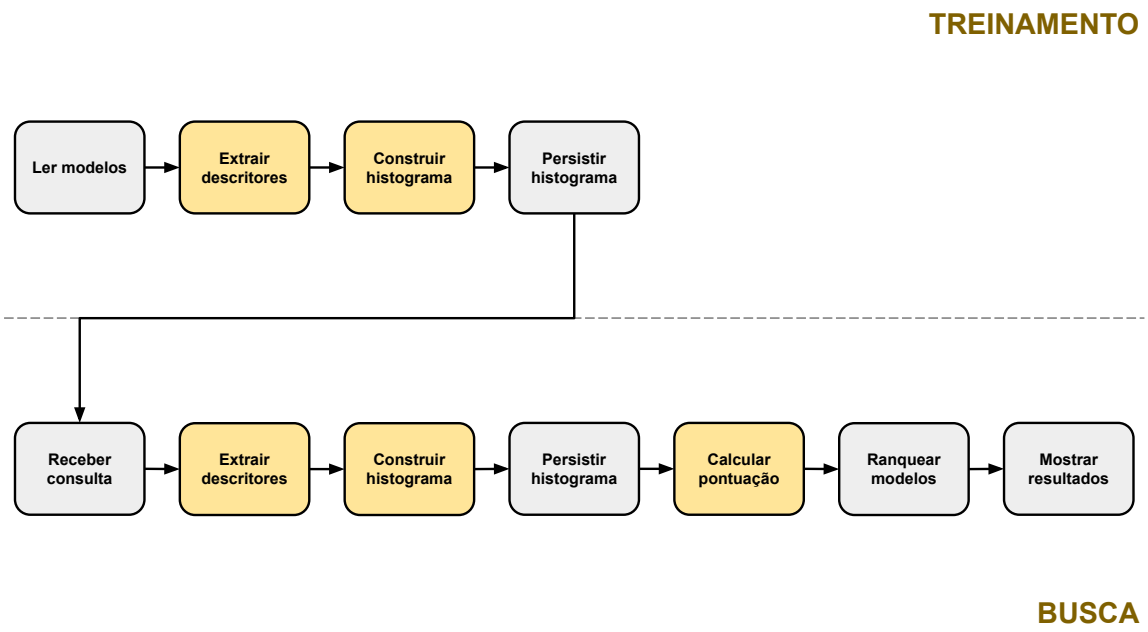
Fonte: (SILVA, 2018).

Esta base de objetos foi escolhida, por contar com objetos simples, com poucos polígonos. O motivo para isto é método de extração de características usado. O grande número de descritores retornados pelo método para modelos complexos (com grande número de polígonos), gera um banco de descritores muito grande, podendo chegar a GB de informação, impossibilitando a execução no ambiente de testes utilizado.

4.2 Algoritmo de Wahl, Hillenbrand e Hirzinger

O algoritmo dos autores Wahl, Hillenbrand e Hirzinger (2003) foi implementado baseado no trabalho original dos mesmos, o qual consiste em duas fases: treinamento e busca, as quais podem ser fragmentadas de acordo com a Figura 10 e conforme detalhado nas subseções 4.2.1 e 4.2.2, respectivamente. A fase de treinamento é a responsável por extrair os descritores para cada modelo da base de dados e gerar os histogramas. Em seguida, a fase de busca tem como papel extrair os descritores do objeto de consulta, montar seu histograma e comparar com os histogramas da base de objetos. Em amarelo estão os pontos chave do processo.

Figura 10 – Visão geral do protótipo.



Fonte: Elaborado pelo autor.

4.2.1 Fase de Treinamento

A fase de treinamento é a base do algoritmo em questão. É nela que o algoritmo consegue aprender sobre os modelos para que seja possível realizar as futuras buscas por modelos semelhantes. O pseudocódigo abaixo apresenta resumidamente esta fase do algoritmo de Wahl, Hillenbrand e Hirzinger.

Algorithm 1: Função de treinamento do algoritmo

```
arquivos_dos_modelos = listar_diretorio(pasta);
modelos = lista();
histogramas = lista();
escore = lista();
for arquivo in arquivos_dos_modelos do
    modelos = adicionar_modelo(modelos, ler_modelo(arquivo));
for modelo in modelos do
    caracteristicas = extrair_caracteristicas(modelo);
    histograma = construir_histograma(caracteristicas);
    histogramas = adicionar_histograma(histogramas, histograma);
for histograma in histogramas do
    arquivar_histograma(histograma);
```

A primeira etapa da fase de treinamento do algoritmo consiste em ler todos os modelos disponíveis na base de objetos. Cada modelo é representado por um arquivo contendo os vértices e faces dos seus polígonos. Essas informações, após serem lidas, são armazenadas em memória através de duas matrizes para cada modelo: uma para as coordenadas e outra para as faces.

Após ler os arquivos dos modelos e armazenar suas informações em memória, suas características são extraídas utilizando a relação par-*surflets*. Para cada par de *surflets* de um modelo, as características α, β, γ e δ são calculadas utilizando as equações 2.5, 2.6, 2.7 e 2.8. Suas características são então armazenadas em uma lista de características. Cada modelo 3D apresenta a sua própria lista de características.

Depois de extrair as características dos modelos, um histograma é gerado para cada modelo, obedecendo a Equação 2.10. Cada histograma é armazenado em um arquivo com o nome do modelo referente ao histograma.

4.2.2 Fase de Busca

A fase de busca é a responsável por retornar o resultado (ou escore) do modelo de consulta, através da comparação entre histogramas. Este processo pode ser visto no pseudocódigo abaixo.

Algorithm 2: Função de busca do algoritmo

```
arquivo_de_consulta = receber_arquivo();
modelo_de_consulta = ler_modelo(arquivo_de_consulta);
caracteristicas = extrair_caracteristicas(modelo_de_consulta);
histograma_de_consulta = construir_histograma(caracteristicas);
arquivar_histograma(histograma_de_consulta);
escore = lista();
for histograma, arquivo in (histogramas, arquivos_dos_modelos) do
    par_modelo_pontuacao = [arquivo,
     calcular_pontuacao(histograma_de_consulta, histograma)];
    escore = adicionar_ao_escore(par_modelo_pontuacao);
mostrar(ranquear(escore));
```

Na fase de busca, um modelo é passado como consulta e é lido da mesma forma que os demais. Suas características são extraídas e seu histograma é construído e persistido. Apesar da semelhança com a fase de treinamento, a fase de busca apresenta mais três passos importantes, sendo eles: calcular a pontuação dos histogramas, ranquear os modelos e mostrar os resultados.

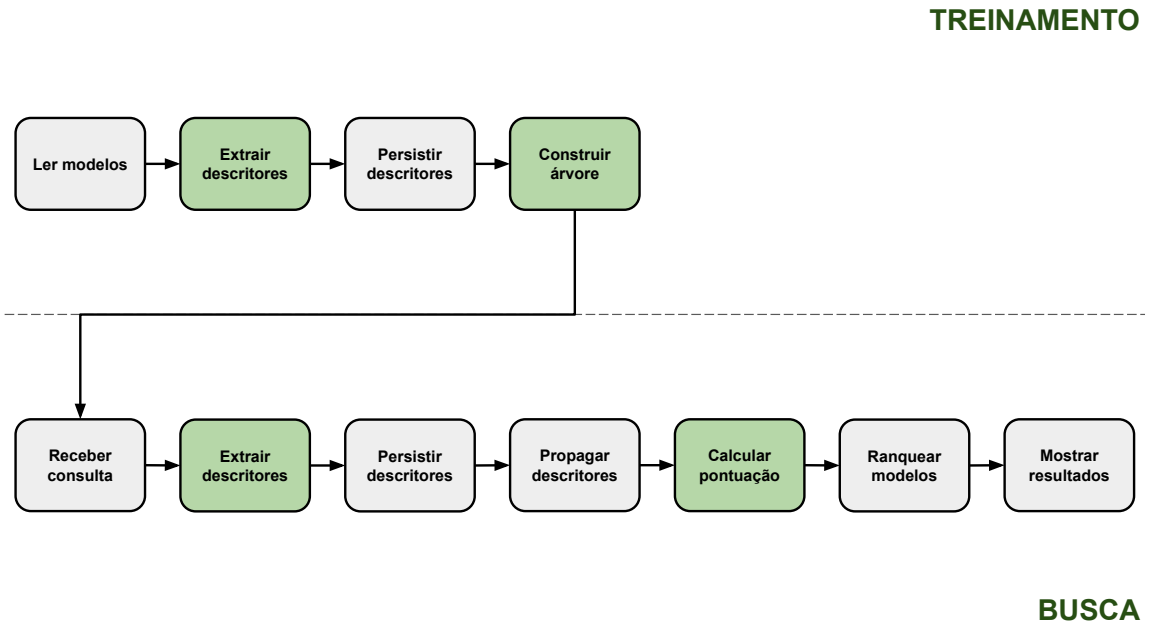
O cálculo da pontuação é feito comparando dois histogramas por uma medida de distância. Os autores Wahl, Hillenbrand e Hirzinger (2003) utilizam uma série de métodos para comparar os histogramas, porém, para quesito de comparação entre os algoritmos implementados, somente a dissimilaridade obtida pela Equação 2.14 foi levada em consideração. Sendo assim, a dissimilaridade é encontrada comparando o histograma gerado pelo modelo de consulta com os histogramas gerados na fase de treinamento.

Assim que as dissimilaridades entre todos os modelos da base de objetos e o modelo de consulta são calculadas, um ranqueamento é feito para descobrir os modelos mais semelhantes ao modelo de consulta. Só então o algoritmo apresenta o resultado obtido.

4.3 Método Proposto

Como mostra a Figura 11, o método também apresenta as fases de treinamento e busca, porém com diferenças sutis com relação ao método de Wahl, Hillenbrand e Hirzinger. Basicamente, no método proposto, a fase de treinamento serve para extrair as características dos modelos da base de dados, persistir essas informações e construir a árvore de vocabulários. Já a fase de busca, tem como objetivo receber um modelo como exemplo de consulta, extrair suas características e procurar por modelos semelhantes na base de dados, usando a árvore para calcular a dissimilaridade. Em verde estão destacadas as tarefas que foram o foco deste trabalho.

Figura 11 – Visão geral do método proposto.



Fonte: Elaborado pelo autor.

4.3.1 Fase de Treinamento

Durante a fase de treinamento, o método recebe como entrada a base de dados com os modelos que devem ser lidos. Estes, por sua vez, são armazenados em memória da mesma maneira implementada no protótipo.

Todos os modelos que foram lidos tem seus descritores calculados pela relação par-*surflets* e armazenados em listas. Esse método foi escolhido para a extração de características, pois utiliza as relações geométricas dos modelos, que são a maior unidade independente identificada para descrever características parciais de um objeto. Os valores das listas são normalizadas e gravadas em arquivos separados, facilitando um futuro treinamento para os mesmos modelos.

Após os descritores terem sido extraídos dos modelos, uma lista, contendo todos os descritores, é criada e passada como entrada para a construção da árvore de vocabulários. O nodo raiz da árvore é iniciado contendo a lista inteira, e seus k filhos são iniciados a partir de fragmentos da lista principal extraídos a partir de um algoritmo de *clustering*. A construção da árvore continua até que um nível l seja alcançado ou até que o número de descritores seja insuficiente para o algoritmo de *clustering*.

4.3.2 Fase de Busca

A fase de busca só pode ser iniciada após a construção da árvore de vocabulários e deve receber como entrada um modelo de consulta. O modelo de consulta tem seus

descritores extraídos e armazenados do mesmo modo que os modelos da base de dados. É importante enfatizar que, caso o modelo já tenha seus descritores armazenados, não é necessário passar por esse processo novamente. Logo após, os descritores são propagados pelos nodos da árvore, calculando suas pontuações de acordo com as equações 2.11, 2.12 e 2.13.

O ranqueamento dos modelos é feito através dos vetores gerados pelas pontuações alcançadas pelos modelos em cada nodo da árvore. Para a construção do vetor, os nodos são visitados através de uma busca em largura. Os modelos são ordenados de acordo com a dissimilaridade do seu vetor de pontuação com o vetor de pontuação do modelo de consulta. Quanto menor o valor da dissimilaridade, mais similares os modelos são e, consequentemente, mais perto do início da fila.

4.3.3 Algoritmo Desenvolvido

O algoritmo foi desenvolvido com o linguagem de programação Python. O motivo da escolha desta linguagem se dá ao fato dela ser de fácil implementação, com muitas abstrações em alto nível e diversos módulos criados pela comunidade, o que proveem um rápido desenvolvimento. Este algoritmo foi implementado com o objetivo de validar este trabalho, gerando os resultados que foram analisados no Capítulo 5. Cada passo do algoritmo, junto com o seu respectivo pseudocódigo logo em seguida, são apresentados abaixo.

O algoritmo começa tendo como entrada uma *string* contendo o caminho do diretório com os modelos da base de dados. Através dessa *string*, a lista com os modelos é obtida e as suas características são calculadas e armazenadas em vetores, sendo o primeiro elemento de cada vetor o nome do arquivo de origem do objeto 3D.

Algorithm 3: Função de treinamento do algoritmo proposto

```
caminho = ler_entrada();
lista_de_arquivos = listar_diretorio(caminho);
arquivos_invertidos = dicionario();
lista_de_caracteristicas = lista();
for arquivo in lista_de_arquivos do
    modelo = ler_modelo(arquivo);
    caracteristicas, arquivos_invertidos = extrair_caracteristicas(modelo,
     arquivos_invertidos);
    arquivar_caracteristicas(caracteristicas);
    lista_de_caracteristicas =
     adicionar_caracteristicas(lista_de_caracteristicas, caracteristicas);
lista_de_caracteristicas = unificar_listas(lista_de_caracteristicas);
arvore = contruir_arvore(lista_de_caracteristicas)
```

Durante o processo de extração das características, um dicionário (um tipo de estrutura de dados do Python) é utilizado como arquivos invertidos, usando como índices *IDs* gerados pelos descritores e armazenando o nome dos arquivos onde esses descritores aparecem. O objetivo é diminuir o espaço de armazenamento e aumentar a velocidade de consulta, uma vez que a quantidade de modelos é muitas vezes menor que a quantidade de descritores.

Algorithm 4: Função de construção dos arquivos invertidos

```
for caracteristica in caracteristicas do
    indice = id(caracteristica);
    arquivos_invertidos = adicionar_arquivo(arquivos_invertidos,
     arquivo_do_modelo, indice);
```

A etapa de treinamento é finalizado quando os vetores de características, extraídos dos objetos 3D, são unificados em uma única lista de descritores. A lista é então passada para a construção da árvore que utiliza o método de *K-Means* como algoritmo de *clustering*, passando para a etapa de busca.

A etapa de busca recebe o nome de um arquivo contendo as características de um modelo como entrada. Isso é feito para agilizar o processo, já que o modelo não precisa recalcular seus descritores. Para o mesmo propósito, ao invés de propagar seus descritores pela árvore, são utilizadas as pontuações geradas na criação da árvore de vocabulários - como os descritores do objeto serão sempre iguais, eles sempre passaram pelos mesmos nodos quando propagados, obtendo a mesma pontuação. Por fim, os vetores de pontuação são gerados usando as pontuações e comparados para ranquear os modelos.

Algorithm 5: Função de busca do algoritmo proposto

```
arquivo_de_consulta = receber_arquivo();
caracteristicas= ler_caracteristicas(arquivo_de_consulta);
vetor_de_consulta, vetores = construir_vetor(arvore, caracteristicas);
escore = lista();
for vetor, arquivo in (vetores, arquivos_dos_modelos) do
    par_modelo_pontuacao = [arquivo,
     calcular_pontuacao(vetor_de_consulta, vetor)];
    escore = adicionar_ao_escore(par_modelo_pontuacao);
mostrar(ranquear(escore));
```

5 RESULTADOS

Neste capítulo são apresentados os resultados obtidos através da execução dos algoritmos implementados e discutidos no Capítulo 4 deste trabalho. Para realizar os testes encontrados neste capítulo, o ambiente adotado foi: computador com processador Core i5, 2.2 GHz, de 5ª geração e 8 GB de memória RAM. Os critérios adotados para os testes são vistos na seção 5.1 e a análise dos resultados utilizando estes critérios é discutida na seção 5.2.

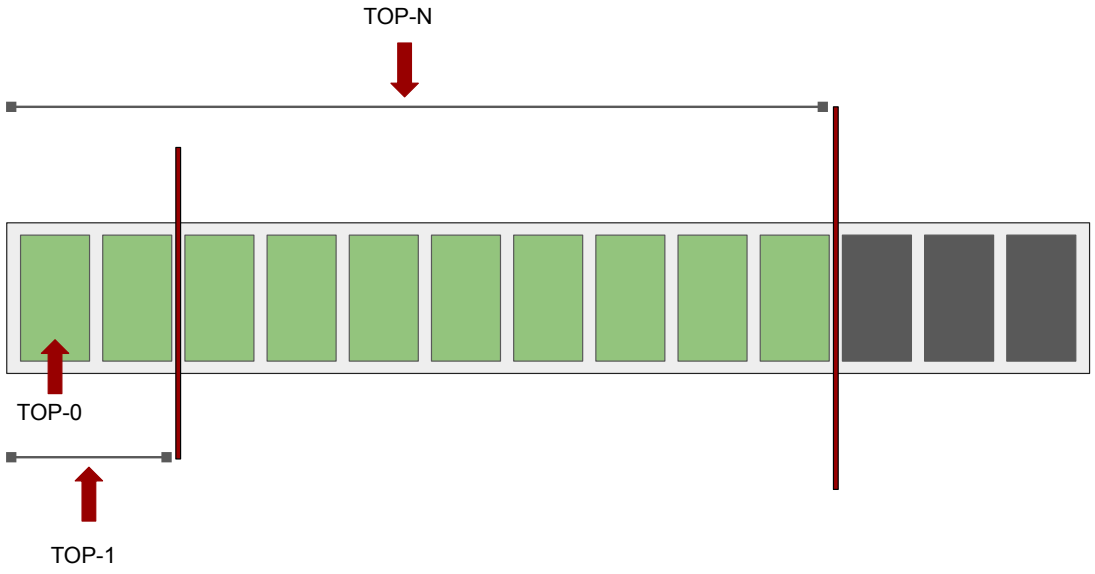
5.1 Critérios Adotados para a Porcentagem de Acertos

Três critérios foram usados para obter a porcentagem de acertos do algoritmo, são eles: TOP-0, TOP-1 e TOP-N. Estes são uma adaptação dos critérios adotados por Paz (2018), sendo ilustrados na Figura 12. O TOP-0 representa a porcentagem de vezes que o algoritmo conseguiu encontrar o próprio modelo de consulta na base de dados, basicamente testando se erros grosseiros não foram cometidos e se o algoritmo funciona minimamente. O TOP-1 é a quantidade de vezes que um modelo semelhante apareceu nas duas primeiras posições do vetor de resultado. Já o TOP-N mostra a porcentagem de vezes que o algoritmo acertou todos os objetos semelhantes nas n primeiras posições do vetor de resultado, sendo n o número de objetos semelhantes presente na base. O pseudocódigo a seguir mostra como cada um desses três critérios são calculados, sendo o escore a representação do vetor de resultado.

Algorithm 6: Função de cálculo das porcentagens dos resultados

```
TOP_0 = 0;
TOP_1 = 0;
TOP_N = 0;
n = numero_de_objetos_semelhantes;
for modelo_semelhante in modelos_semelhantes do
    if modelo_semelhante == escore[0] then
        TOP_0 += 1;
    end
    if modelo_semelhante in escore[: 2] then
        TOP_1 += 1;
    end
    if modelo_semelhante in escore[: n] then
        TOP_N += 1;
    end
end
TOP_0_PERCENT = TOP_0 / quantidade_objetos * 100;
TOP_1_PERCENT = TOP_1 / quantidade_objetos * 100 / 2;
TOP_N_PERCENT = TOP_N / quantidade_objetos * 100 / n;
```

Figura 12 – Estretégia para avaliar a precisão da aplicação.

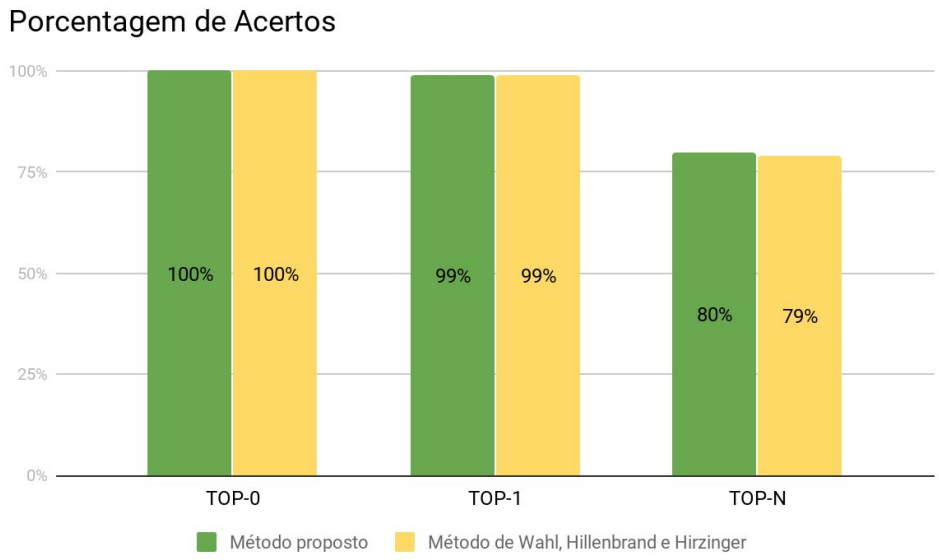


Fonte: Elaborado pelo autor.

5.2 Análise dos Resultados na Base de Objetos 3

Os números mostrados na Figura 13 foram obtidos utilizando $k = 4$ e $l = 6$ para o método proposto. Nela é possível observar a taxa de acertos para todas as métricas de porcentagem de acertos. Em 100% das consultas realizadas pelos dois algoritmos, modelos iguais foram retornados no topo do escore, mostrando que os algoritmos conseguem achar o próprio modelo de consulta na base de dados. A segunda métrica também apresentou resultados iguais para ambos os algoritmos. A única métrica que apresentou diferença nos resultados foi a TOP-N, onde o algoritmo proposto apresentou uma melhora de 1% comparado com o algoritmo presente na literatura, com $n = 10$.

Figura 13 – Porcentagem de acertos obtidos pelos algoritmos.

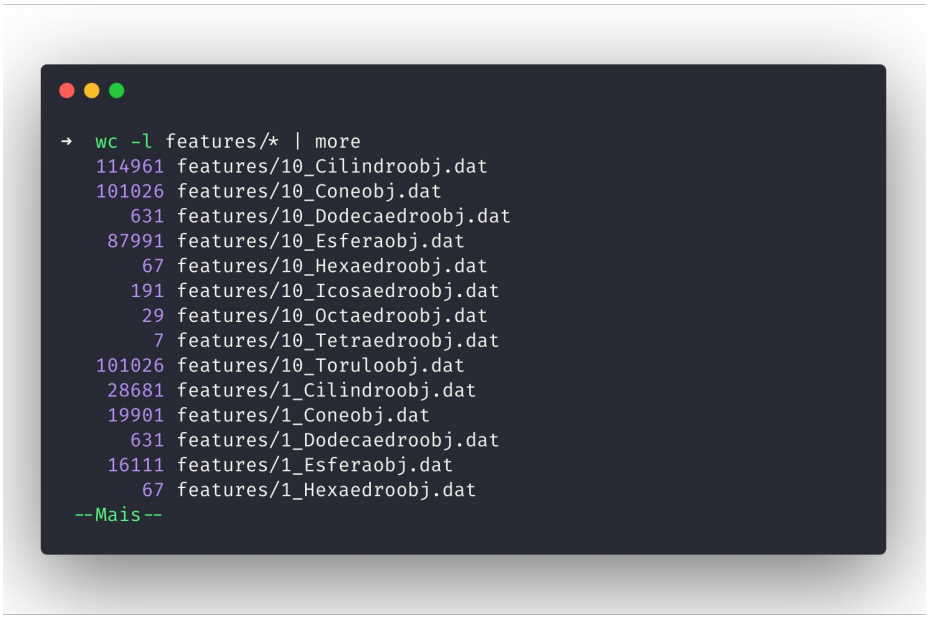


Fonte: Elaborado pelo autor.

Uma possível explicação para a porcentagem de 80% pode ser a baixa quantidade de descritores em alguns modelos, devido à sua baixa complexidade. Isso pode ser observado na Figura 14, onde são mostradas as quantidades de linhas de alguns dos arquivos que guardam as características dos modelos, sendo que cada linha representa uma característica, com exceção da primeira. Nesta imagem é possível observar alguns arquivos com mais de 100000 características e outros com apenas 6 (7 - 1). Isso se dá ao fato da quantidade de polígonos ter uma diferença muito significativa entre os modelos, consequentemente, uma diferença considerável na quantidade de pares de *surflets*.

Essa teoria ganha força quando analisamos os resultados das consultas para dois objetos diferentes: tetraedro e toróide. O primeiro modelo usado como consulta foi esco-

Figura 14 – Número de linhas dos arquivos de descritores.



Fonte: Elaborado pelo autor.

lhido por sua baixa quantidade de descritores, apresentando apenas 6. O segundo modelo foi escolhido pelo motivo oposto, apresentando 101025 características extraídas.

Na Figura 15 é possível visualizar o resultado do algoritmo de Wahl, Hillenbrand e Hirzinger para os dois modelos escolhidos. O modelo usado como consulta aparece como uma *string*, seguido de uma seta que aponta para o vetor de resultado com dez posições, que apresenta as *strings* dos arquivos que contém as características dos modelos. O vetor é mostrado com dez posições por haver dez modelos semelhantes em cada classe. As duas consultas são apresentadas em sequência, separadas por duas quebras de linha. Na figura, podemos observar que o algoritmo conseguiu retornar todos os modelos das classes dos modelos de consulta, mostrando que a diferença na quantidade de descritores não é o fator principal para o resultado desejado.

A Figura 16 apresenta as mesmas consultas, seguindo o mesmo formato, porém utilizando o algoritmo do método proposto. O resultado obtido pelo modelo com menos características apresentou somente dois modelos da mesma da classe de objetos. Porém, a consulta realizada para o objeto com grande número de características apresentou todos os objetos da classe. As duas consultas também podem ser visualizadas em forma gráfica na imagem Figura 17.

Após a análise dos resultados mostrados anteriormente, é possível observar que modelos com poucos descritores interferem negativamente no resultado do método proposto, porém não interferem no algoritmo de Wahl, Hillenbrand e Hirzinger. Para comprovar esse

Figura 15 – Saída apresentada pelo algoritmo de Wahl, Hillenbrand e Hirzinger.

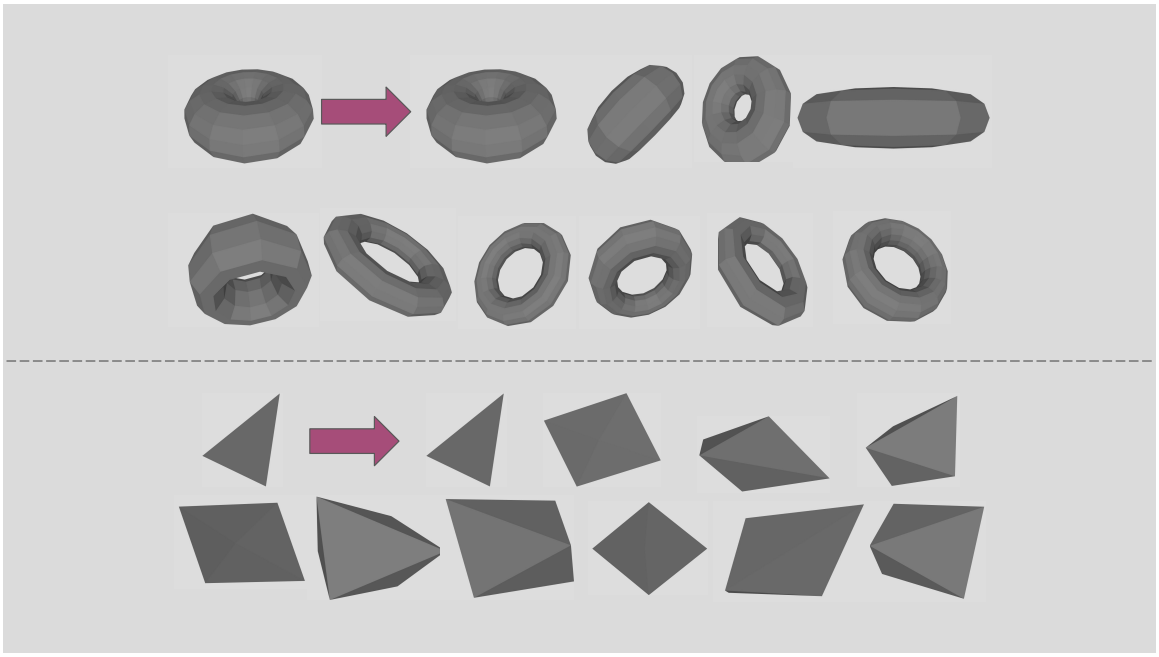
Fonte: Elaborado pelo autor.

Figura 16 – Saída apresentada pelo algoritmo do método proposto.

Fonte: Elaborado pelo autor.

raciocínio, um subconjunto de objetos foi criado, excluindo as cinco classes de objetos que apresentaram os menores números de descritores em seus modelos. O critério utilizado

Figura 17 – Visualização do resultado obtido na Figura 16 de forma gráfica



Fonte: Elaborado pelo autor.

para isto foi: excluir as classes de objetos que apresentem modelos com menos de 1000 características. As classes excluídas foram: tetraedro, octaedro, hexaedro, dodecaedro e icosaedro.

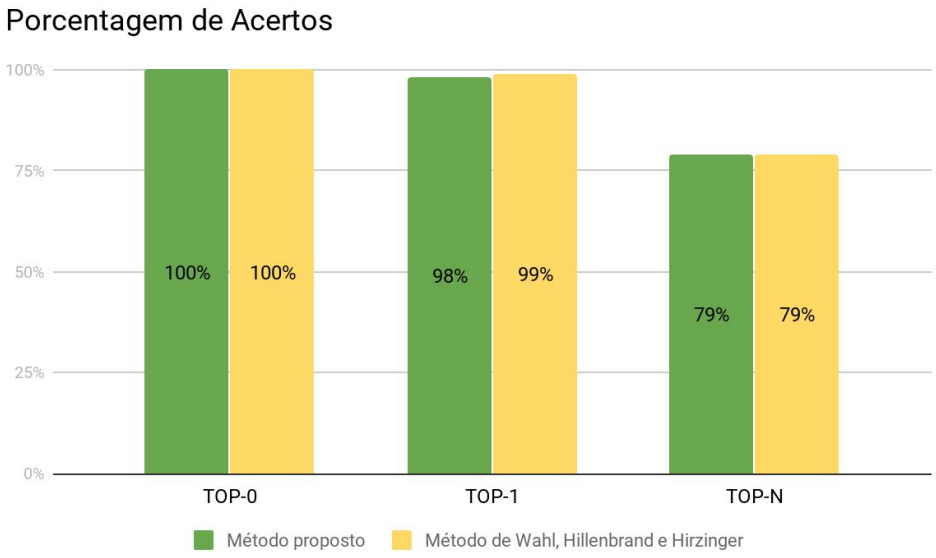
O novo conjunto de objetos apresenta 40 modelos, todos com mais de 1000 características. Na Figura 18 são mostrados os resultados obtidos pelos algoritmos utilizando a nova base de dados. Os parâmetros utilizados foram os mesmos do teste anterior. Apesar da mudança do banco de dados, excluindo as classes que apresentavam poucas características, os resultados obtidos foram inferiores. Isso aconteceu por causa do *overfitting* alcançado pelos parâmetros utilizados. Intuitivamente, o esperado seria o aumento das porcentagens, porém os parâmetros k e l utilizados no primeiro teste utilizavam como base o número de características extraídas do banco de dados anterior.

Segundo Paz (2018) os parâmetros k e l para que a árvore de vocabulários consiga atingir resultados satisfatórios em seu algoritmo podem ser definidos por

$$DF = \frac{nd}{k^L}, \tag{5.1}$$

onde k é o grau da árvore, L é a quantidade de níveis, nd é a quantidade de descritores extraídos do conjunto de objetos e DF é o valor estatístico que permite aproximar a quantidade ótima de folhas para uma dada quantidade de descritores. Apesar desta equação não apresentar bons resultados para o algoritmo proposto, a relação entre a quantidade de descritores e a quantidade de folhas da árvore se mantém, por este motivo, novos parâmetros devem ser utilizados para a nova base de objetos.

Figura 18 – Porcentagem de acertos obtidos pelos algoritmos através do subconjunto de modelos gerado.



Fonte: Elaborado pelo autor.

Levando em consideração a relação entre a quantidade de características e a quantidade de folhas, um novo teste foi realizado utilizando como parâmetros $k = 3$ e $l = 4$, apresentado na Figura 19. Neste novo teste os dois algoritmos apresentaram 100% de acertos nas duas primeiras métricas, novamente apresentando resultados iguais para elas. Entretanto, na última métrica, a diferença de 1% do primeiro teste aumentou para 5%, dando mais força ao questionamento levantado anteriormente. Contudo, ainda não é possível ter absoluta certeza que somente o número de características foi o responsável pelo aumento na porcentagem de acertos, uma vez que a quantidade de modelos também mudou. Por este motivo mais testes se fazem necessários para comprovar a análise feita através dos resultados obtidos neste trabalho.

Infelizmente, por ainda não existir uma equação capaz de gerar parâmetros ideias para o algoritmo proposto, os testes foram realizados com parâmetros gerados por força bruta, testando os possíveis valores de k e l até que o algoritmo apresentasse *underfitting* e *overfitting*. Após os inúmeros testes, os parâmetros que apresentaram os melhores resultados foram selecionados.

Figura 19 – Porcentagem de acertos obtidos pelos algoritmos através do subconjunto de modelos gerado utilizando como parâmetros $k = 3$ e $l = 4$.

Fonte: Elaborado pelo autor.

6 CONSIDERAÇÕES FINAIS

Com o presente trabalho foi possível constatar a importância do desenvolvimento de métodos eficazes para recuperação de modelos 3D, apresentando as definições referentes à fundamentação teórica e aos métodos relacionados que foram empregados para o desenvolvimento deste trabalho. Com base na revisão da literatura realizada, também foi possível constatar que não existem métodos com entrada baseada em modelos 3D que apresentam como forma de indexação de características a árvore de vocabulários, o que é proposto neste trabalho.

Um algoritmo foi implementado tendo como base o método proposto de recuperação de objetos 3D, tendo seus resultados analisados e discutidos por este trabalho. Apesar das dificuldades encontradas por conta da limitação de memória, os resultados apresentados são promissores, com valores de porcentagem de acerto acima de 80%, comprovando que é possível, através de métricas, recuperar os modelos semelhantes ao modelo de exemplo.

Como principal ponto de melhoria para trabalhos futuros, podemos citar a etapa de extração de características. Nela, pode-se investigar os efeitos de selecionar os melhores pontos para extração de características, como proposto por Mian, Bennamoun e Owens (2010), reduzindo o tempo e o consumo de memória na extração e indexação de características, permitindo a utilização de bancos de objetos mais complexos.

REFERÊNCIAS

ANDERSON, B. J. et al. Adapting k-medians to generate normalized cluster centers. In: SIAM. **Proceedings of the 2006 SIAM International Conference on Data Mining**. [S.l.], 2006. p. 165–175. Citado na página 36.

CAO, D.; YANG, B. An improved k-medoids clustering algorithm. In: IEEE. **2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)**. [S.l.], 2010. v. 3, p. 132–135. Citado na página 36.

CISCO. **Cisco Visual Networking Index: Forecast and Methodology, 2016–2021**. [S.l.], 2017. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf>. Acesso em: 12.05.2019. Citado 2 vezes nas páginas 28 e 29.

DAVIES, E. R. **Computer and machine vision: theory, algorithms, practicalities**. [S.l.]: Academic Press, 2012. Citado na página 27.

DESJARDINS, J. **What Happens in an Internet Minute in 2019?** [S.l.], 2019. Disponível em: <https://www.visualcapitalist.com/what-happens-in-an-internet-minute-in-2019/>. Acesso em: 12.05.2019. Citado na página 29.

GRANA, C. et al. A fast approach for integrating orb descriptors in the bag of words model. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Multimedia Content and Mobile Devices**. [S.l.], 2013. v. 8667, p. 866709. Citado 2 vezes nas páginas 36 e 43.

IYER, N. et al. Three-dimensional shape searching: state-of-the-art review and future trends. **Computer-Aided Design**, v. 37, n. 5, p. 509 – 530, 2005. ISSN 0010-4485. Geometric Modeling and Processing 2004. Disponível em: <http://www.sciencedirect.com/science/article/pii/S001044850400140X>. Citado na página 30.

JAIN, A. K. Data clustering: 50 years beyond k-means. **Pattern Recognition Letters**, v. 31, n. 8, p. 651 – 666, 2010. ISSN 0167-8655. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR). Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167865509002323>. Citado 2 vezes nas páginas 36 e 43.

JASTROW, J. The mind's eye. **Popular Science Monthly**, 1899. Citado na página 30.

LOGOGLU, K. B.; KALKAN, S.; TEMIZEL, A. Cospair: colored histograms of spatial concentric surflet-pairs for 3d object recognition. **Robotics and Autonomous Systems**, Elsevier, v. 75, p. 558–570, 2016. Citado na página 44.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**. [S.l.], 1967. v. 1, n. 14, p. 281–297. Citado na página 37.

MARS, D. E. et al. Geometry-based ranking for mobile 3d visual search using hierarchically structured multi-view features. In: **2015 IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2015. p. 3077–3081. Citado na página 43.

MIAN, A.; BENNAMOUN, M.; OWENS, R. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. **International Journal of Computer Vision**, Springer, v. 89, n. 2-3, p. 348–361, 2010. Citado 2 vezes nas páginas 44 e 63.

NISTER, D.; STEWENIUS, H. Scalable recognition with a vocabulary tree. In: **2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)**. [S.l.: s.n.], 2006. v. 2, p. 2161–2168. ISSN 1063-6919. Citado 4 vezes nas páginas 31, 38, 42 e 43.

PAZ, G. L. Monografia (Bacharel em Ciência da Computação), **OpenVT: Aplicação Escalável de Código Aberto para Busca de Imagens Semelhantes Utilizando Árvore de Vocabulário**. 2018. Disponível em: <http://dspace.unipampa.edu.br:8080/jspui/handle/riu/3330>. Acesso em: 10.05.2019. Citado 6 vezes nas páginas 31, 41, 42, 43, 55 e 60.

PERRONNIN, F.; DANCE, C. Fisher kernels on visual vocabularies for image categorization. In: **2007 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2007. p. 1–8. ISSN 1063-6919. Citado na página 43.

RUBLEE, E. et al. Orb: An efficient alternative to sift or surf. In: **2011 International Conference on Computer Vision**. [S.l.: s.n.], 2011. v. 11, n. 1, p. 2. Citado na página 43.

SAVELONAS, M. A.; PRATIKAKIS, I.; SFIKAS, K. Fisher encoding of differential fast point feature histograms for partial 3d object retrieval. **Pattern Recognition**, Elsevier, v. 55, p. 114–124, 2016. Citado na página 43.

SILVA, S. A. da. Monografia (Bacharel em Ciência da Computação), **Recuperação de Objetos 3D Baseada em Relações Geométricas entre *Surflets***. 2018. Disponível em: <http://www.if.ufrgs.br/~thielo/sherlontcc.pdf>. Acesso em: 17.05.2019. Citado 6 vezes nas páginas 17, 41, 42, 47, 48 e 73.

SIVIC, J.; ZISSERMAN, A. Video google: A text retrieval approach to object matching in videos. In: IEEE. **Proceedings Ninth IEEE International Conference on Computer Vision**. [S.l.], 2003. p. 1470. Citado na página 38.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. [S.l.], 2010. Disponível em: <http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf>. Acesso em: 12.05.2019. Citado na página 33.

TANGELDER, J. W. H.; VELTKAMP, R. C. A survey of content based 3d shape retrieval methods. In: **Proceedings Shape Modeling Applications, 2004.** [S.l.: s.n.], 2004. p. 145–156. Citado 3 vezes nas páginas 27, 30 e 31.

WAHL, E.; HILLENBRAND, U.; HIRZINGER, G. Surflet-pair-relation histograms: a statistical 3d-shape representation for rapid classification. In: **Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003.**

Proceedings. [S.l.: s.n.], 2003. p. 474–481. Citado 8 vezes nas páginas 31, 35, 37, 38, 42, 43, 49 e 51.

ZOU, K.; ZHANG, Q. Research progresses and trends of content based 3d model retrieval. In: **2018 Chinese Control And Decision Conference (CCDC)**. [S.l.: s.n.], 2018. p. 3346–3351. ISSN 1948-9447. Citado na página 29.

Apêndices

APÊNDICE A – REPOSITÓRIO UTILIZADO

1 - Clone o repositório pelo link:

```
$ git clone https://github.com/161150744/RecObj3DVocabTree.git
```

2 - Instale as dependencias:

```
$ pip install -r requirements.txt
```

3 - Para obter informações de como utilizar:

```
$ python3 main.py -h
```

4 - Base de Objetos utilizada:

```
$ ls ./algoritmo_proposto/objs/
```

APÊNDICE B – BASE DE OBJETOS 3

Tabela 1 – Base de Objetos 3 - 90 Objetos Regulares criados por Silva (2018)

Classe	Total	Objetos
1_Dodecaedro	10	1_Dodecaedro 2_Dodecaedro 3_Dodecaedro 4_Dodecaedro 5_Dodecaedro 6_Dodecaedro 7_Dodecaedro 8_Dodecaedro 9_Dodecaedro 10_Dodecaedro
2_Hexaedro	10	1_Hexaedro 2_Hexaedro 3_Hexaedro 4_Hexaedro 5_Hexaedro 6_Hexaedro 7_Hexaedro 8_Hexaedro 9_Hexaedro 10_Hexaedro
3_Icosaedro	10	1_Icosaedro 2_Icosaedro 3_Icosaedro 4_Icosaedro 5_Icosaedro 6_Icosaedro 7_Icosaedro 8_Icosaedro 9_Icosaedro 10_Icosaedro
4_Octaedro	10	1_Octaedro 2_Octaedro 3_Octaedro 4_Octaedro 5_Octaedro 6_Octaedro 7_Octaedro 8_Octaedro 9_Octaedro 10_Octaedro
5_Tetraedro	10	1_Tetraedro 2_Tetraedro 3_Tetraedro 4_Tetraedro 5_Tetraedro 6_Tetraedro 7_Tetraedro 8_Tetraedro 9_Tetraedro 10_Tetraedro
6_Cilindro	10	1_Cilindro 2_Cilindro 3_Cilindro 4_Cilindro 5_Cilindro 6_Cilindro 7_Cilindro 8_Cilindro 9_Cilindro 10_Cilindro
7_Cone	10	1_Cone 2_Cone 3_Cone 4_Cone 5_Cone 6_Cone 7_Cone 8_Cone 9_Cone 10_Cone
8_Esfera	10	1_Esfera 2_Esfera 3_Esfera 4_Esfera 5_Esfera 6_Esfera 7_Esfera 8_Esfera 9_Esfera 10_Esfera
9_Toroide	10	1_Toroide 2_Toroide 3_Toroide 4_Toroide 5_Toroide 6_Toroide 7_Toroide 8_Toroide 9_Toroide 10_Toroide

APÊNDICE C – SUBCONJUNTO UTILIZADO DA BASE DE OBJETOS 3

Tabela 2 – Subconjunto da Base de Objetos 3 com 40 Objetos

Classe	Total	Objetos
6_Cilindro	10	1_Cilindro 2_Cilindro 3_Cilindro 4_Cilindro 5_Cilindro 6_Cilindro 7_Cilindro 8_Cilindro 9_Cilindro 10_Cilindro
7_Cone	10	1_Cone 2_Cone 3_Cone 4_Cone 5_Cone 6_Cone 7_Cone 8_Cone 9_Cone 10_Cone
8_Esfera	10	1_Esfera 2_Esfera 3_Esfera 4_Esfera 5_Esfera 6_Esfera 7_Esfera 8_Esfera 9_Esfera 10_Esfera
9_Toroide	10	1_Toroide 2_Toroide 3_Toroide 4_Toroide 5_Toroide 6_Toroide 7_Toroide 8_Toroide 9_Toroide 10_Toroide