Anderson Bessa da Costa

Aprendizado de Máquina em Representações Tridimensionais

Campo Grande - MS, Brasil

10 de Maio de 2013

Anderson Bessa da Costa

Aprendizado de Máquina em Representações Tridimensionais

Dissertação de Mestrado em Ciência da Computação pela Universidade Federal de Mato Grosso do Sul.

Orientador:
Prof. Dr. Edson Takashi Matsubara

FACULDADE DE COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL

Campo Grande - MS, Brasil

10 de Maio de 2013

Dissertação de Mestrado em Ciência da Computação pela Universidade Federal de Mato Grosso do Sul sob o título *"Aprendizado de Máquina em Representações Tridimensionais"*,

defendido por Anderson Bessa da Costa e aprovado em 10 de Maio de 2013, em Campo Grande, estado de Mato Grosso do Sul, pela banca examinadora constituída pelos professores:

Prof. Dr. Edson Takashi Matsubara
Orientador

Prof. Dr. Gedson Faria
Universidade Federal de Mato Grosso do Sul

Prof. Dr. Hemerson Pistori
Universidade Católica Dom Bosco

Dedicatória

"*Aos pais, familiares e amigos...*

Agradecimentos

Primeiro antes de tudo, gostaria de agradecer àqueles que me proporcionaram a oportunidade de cursar o Mestrado. Esta é uma experiência enriquecedora e bastante árdua, onde aprendemos a enfrentar e buscar soluções diariamente, necessitando sempre de alguém do nosso lado.

Gostaria de agradecer à Deus, por tudo que me foi dado. Pela vida, saúde, calma. Sem a sua ajuda tudo seria mais difícil.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior pela concessão da Bolsa CAPES-REUNI, sem a qual a realização deste sonho seria impossível.

Aos meus pais Danilo e Vera pelos momentos que me proporcionaram. A vocês, minha eterna gratidão.

À minha família pelo tranquilidade e conforto proporcionado.

Ao Prof. Dr. Edson Takashi Matsubara, com quem sua ajuda ultrapassou a esfera acadêmica e acabou se tornando um grande amigo. Seus conselhos e a sua grande ajuda foram de grande importância não apenas para a conclusão deste mestrado, mas para meu crescimento como pessoa. Obrigado por me fazer enxergar além do óbvio e valorizar as coisas importantes.

Ao pessoal do LIA (Laboratório de Inteligência Artificial), que acabaram se tornando minha segunda família, minha segunda casa. Todos vocês me ajudaram bastante sempre que precisei .. não vou conseguir citar todos os nomes, mas gostaria de falar de alguns (não necessariamente os mais importantes): Carlos Emílio, Hudson Fujikawa, Eduardo Theodo, Adauto Ferreira e todas as outras pessoas. Obrigado. Vocês foram muito parceiros na hora em que mais precisava.

Enfim, a todos aqueles que colaboraram de alguma forma ao longo desta trajetória. Me perdoem aqueles que não citei .. se você está lendo este trabalho você é muito especial para mim.

"*A vida só pode ser compreendida olhando-se para trás; mas só pode ser vivida olhando-se para frente.*"

Soren Kierkegaard

Resumo

Aprendizado consiste em adquirir novos, ou modificar os existentes, conhecimentos, comportamentos, habilidades, valores ou preferências e pode envolver a síntese de diferentes tipos de informação. O aprendizado está fortemente relacionado à representação do conhecimento. Uma representação de conhecimento está ligada a capacidade de se inferir, ou seja, aprender. Uma representação de dados bastante presente nos últimos tempos se refere à representação tridimensional, associada fortemente a imagens *3D*. Neste trabalho apresenta-se um estudo sobre aprendizado *3D* com base em dois estudos de caso, imagens *3D* adquiridas a partir do *Kinect* e movimentos adquirido a partir dos dados do *wiimote*. A visão humana sempre despertou um grande fascínio por ser considerada um dos sensores humanos mais ricos em informação. Sempre existiu um grande interesse em simular a capacidade de visão do ser humano. Com o surgimento e popularização de sistemas *RGB-D* (*RedGreenBlue - Depth)*, é possível obter informação de profundidade de cada *pixel* de maneira rápida e confiável. Isto permitiu a aproximação da tecnologia atual com a visão humana, que com dois olhos são capazes de estimar a profundidade aproximada de um objeto. Uma imagem capturada de um sistema *RGB-D* é muito mais rica em informação do que uma imagem capturada de uma câmera comum. Estudar como este ganho de informação pode ser utilizado para melhorar a capacidade de representação e reconhecimento de objetos em Inteligência Artificial é um dos objetivos deste projeto. Por meio de uma proposta de um *pipeline* de aprendizado *3D*, desde captura dos dados até algoritmos de reconhecimento, será provido um estudo sobre aprendizado em sistemas *RGB-D*. Assim como o *Kinect*, o nintendo *Wii* também revolucionou a indústria de games graças ao seu controle capaz de reconhecer os movimentos do jogador. Neste trabalho apresenta-se um estudo sobre reconhecimento de padrões em estruturas tridimensionais com o objetivo de explicitar que com a utilização da técnica de aprendizado de máquina juntamente com a representação adequada do problema é possível atingir altas taxas de acerto com um baixo tempo de processamento, o que foi realizado com sucesso.

Palavras-chave: Visão Computacional, Aprendizado 3D, Segmentação de Imagens

Abstract

Learning is acquiring new or modifying existing knowledge, behaviors, skills, values, or preferences and may involve synthesizing different types of information. Learning is strongly related to knowledge representation. A knowledge representation is linked to the ability to infer, or learn. A data representation very seen nowdays is tridimensional representation, which is strongly associated with 3D images. This work presents a study on 3D learning based on two case studies, 3D images acquired from the Kinect and motion data acquired from the wiimote. Human vision has always aroused a great fascination for being considered one of the richest human sensors. There has always been a great interest in the ability to simulate human vision. With the emergence and popularization of systems RGB-D (RedGreenBlue - Depth), it is possible to obtain depth information of each pixel quickly and reliably. This allowed the approach of current technology with human vision, with two eyes that are able to estimate the approximate depth of an object. A captured image of a system RGB-D is much richer in information than a captured image of an ordinary camera. Study how this information gain can be used to improve the ability of representation and recognition of objects in Artificial Intelligence is one of the goals of this project. Through a proposal of a 3D pipeline learning, from data capture to recognition algorithms, will be provided a study on learning systems RGB-D. Just as the Kinect, the Nintendo Wii also revolutionized the gaming industry thanks to its control able to recognize the player's movements. This paper presents a study on recognition of patterns in three-dimensional structures with the purpose of clarifying that the use of the technique of machine learning along with adequate representation of the problem is possible to achieve high hit rates with a low processing time, which was successful.

Keywords: Computer Vision, 3D Learning, Image Segmentation

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

1 Introdução

A inteligência dos seres humanos é caracterizada não somente pela aquisição e armazenamento de conhecimento, mas principalmente por sua capacidade de inferir novos conhecimentos (utilizando-se daqueles já adquiridos) e adaptar conhecimentos de acordo com o ambiente e/ou seus objetivos a serem atingidos (BRACHMAN; LEVESQUE, 2004). O objetivo deste capítulo consiste em apresentar uma breve introdução e contextualização do trabalho. Na Seção 1.1 são apresentados o contexto e a motivação, na Seção 1.2 são apresentados os objetivos e, por fim, na Seção 1.3 são apresentadas a estrutura e a organização.

1.1 Contexto

Representação de conhecimento é uma área existente dentro da Inteligência Artificial preocupada em como representar o conhecimento de forma a facilitar a representação e capacidade de programas inferirem novos conhecimentos com base no conhecimento existente (BRACHMAN; LEVESQUE, 2004). Em outras palavras, esta área de conhecimento está preocupada em como os agentes utilizam o que sabem para decidir o que fazer. Segundo Brachman e Levesque (2004), existem três conceitos básicos nesta área:

Conhecimento: consiste em uma relação entre o conhecedor e uma proposição. Por exemplo: "João sabe que Maria irá vir a festa". A proposição é expressa por uma sentença declarativa simples. O principal desafio do conhecimento estaria na natureza das proposições;

Representação: dentro da área de conhecimento, representação pode ser vista como uma relação entre dois domínios, no qual o primeiro domínio estaria "tomando" lugar do segundo domínio. Corresponde ao campo de estudo que aborda a utilização de símbolos formais para representar uma coleção de proposições; e

Raciocínio: corresponde a manipulação formal dos símbolos representando uma coleção de proposições afim de produzir novos conhecimentos, ou seja, inferir novos conhecimentos a partir daqueles já adquiridos.

Conhecimento é um conceito bastante abstrato e difícil de se trabalhar. Normalmente, trabalhos que buscam uma contribuição para a área de Inteligência Artificial trabalham sob duas frentes: representação e raciocínio. Grande parte dos trabalhos se concentram no raciocínio, ou seja, em maneiras de inferir conhecimento a partir dos dados de entrada.

A representação tabular, tabelas atributo-valor, tem sido por muitos anos a principal representação computacional utilizada para armazenar os dados em sistemas computacionais. Observe que os banco de dados transacionais atualmente são compostos principalmente por esta representação.

Entretanto, nem sempre é muito intuitivo representar o mundo real utilizando este formato. Representar uma cadeira como uma linha em uma tabela atributo-valor com os campos cor, comprimento e altura fica muito além da representação real de uma cadeira. Em uma tarefa de aprendizado de máquina, na maior parte das vezes, um objeto é codificado no formato tabular onde muita informação é perdida. Dificilmente alguém conseguiria reconstruir a cadeira utilizando apenas cor, comprimento e altura.

A limitação na obtenção de representações mais ricas está nesse processo de transformação e codificação de objetos e fatos do mundo real para uma representação computacional que normalmente é feita como uma tabela atributo-valor. A transformação é feita pela observação dos fatos e registro em um banco.

Nesta dissertação evita-se a representação de tabelas atributo-valor buscando-se por alternativas mais ricas para a representação de instâncias e exemplos. Mais especificamente, o foco deste trabalho está em representações tridimensionais dos dados.

Por meio de um *framework* de reconhecimento de objetos *3D*, neste trabalho objetiva-se a utilização do conceito de representação de um objeto real em um objeto computacional. A ideia (baseada em orientação a objetos) consiste na crença de acreditar-se que a abstração dos dados para mais alto nível traga melhorias significativas no desenvolvimento e complexidade dos algoritmos de reconhecimento.

Dentro do contexto do *pipeline* destaca-se a proposta de dois algoritmos: *downsampling* e segmentação. Algoritmos de processamento de nuvem de pontos são custosos, tendo a sua complexidade proporcional a quantidade de pontos presentes na nuvem. O algoritmo de *downsampling* consiste em reduzir a quantidade de pontos que representam a nuvem de pontos, com o mínimo de impacto de perda de informação.

O algoritmo de segmentação tem por objetivo "quebrar" a nuvem de pontos em segmentos. Este trabalho apresenta a proposta de um algoritmo de segmentação que utiliza os conceitos de

localidade espacial, geometria e coloração. Ao final da seção de segmentação, é apresentado uma comparação do algoritmo proposto com o presente na PCL (*Euclidean Cluster Extraction*), onde o algoritmo proposto apresenta melhores resultados.

Em reconhecimento de padrões, é realizado um trabalho sobre estruturas naturalmente tridimensionais com a captura dos dados a partir do *wiimote*. São lidos triplas (x, y, z) do acelerômetro do *wiimote* a cada período curto de tempo. Cada movimento então é representado como uma curva 3D. É apresentado um estudo da utilização do KNN juntamente com a medida de distância de DTW. Uma terceira técnica TRKNN é utilizada com o intuito de reduzir a quantidade de exemplos armazenados no KNN.

O equipamento utilizado para aquisição de imagens 3D foi inicialmente sistemas *stereo* e posteriormente sistemas *RGB-D*, especificamente o *Kinect*. Com o lançamento do *Kinect* ocorreu uma grande popularização de câmeras *RGB-D*. Anteriormente, imagens *3D* eram difíceis de serem obtidas. Duas outras configurações bastante comum que possibilitam a obtenção de imagens *3D*, entretanto com algumas limitações são: câmeras *stereo*, exigindo um grande custo computacional (HENGSTLER, 2008), e sistemas de câmeras com *laser*.

Comparado com imagens obtidas de câmeras convencionais, imagens obtidas a partir de sistemas *RGB-D* possuem informação de profundidade de cada *pixel*. Isto representa um ganho a mais de informação. A partir desta informação adicional, buscam-se meios que permitam melhorar a maneira como objetos são representados e consequentemente reconhecidos.

O *Kinect* é conhecido pelo seu uso em conjunto com o videogame *Xbox 360* lançado pela *Microsoft*. A sua utilização é notável no reconhecimento de movimentos, proporcionando uma maior interação entre o jogador e o jogo, possibilitando que jogos antes presentes apenas no imaginário do jogador se tornassem realidade. Além do *Kinect* outro equipamento conhecido na área de jogos utilizado para o mesmo fim é o *wiimote* do *Nintendo Wii*. *Wiimote* é um controle que contêm sensores como giroscópio e acelerômetro. Embora tenham o mesmo objetivo, os dois equipamentos se diferem bastante na maneira como este objetivo é alcançado.

1.2 Objetivos

A proposta deste trabalho consiste em trabalhar com dados que naturalmente são estruturas tridimensionais: imagens adquiridas a partir de sistemas de câmeras *stereo* e *Kinect* e dados do sensores de acelerômetro do *wiimote*.

Busca-se por representações mais ricas do que a representação atributo-valor. Neste traba-

lho é feito estudo sobre as representações de lógica de primeira ordem, lógica proposicional e a representação por objetos. Esta última foi em parte aplicada ao trabalho para a representação de objetos tridimensionais.

Inicialmente os dados devem ser capturados. As imagens são capturadas utilizando câmeras *stereo* e posteriormente o *Kinect*. É proposto um modelo de representação de objetos inspirado nos conceitos de orientação a objetos, tais como: relacionamentos, comportamentos e características. Por meio de um *pipeline* de aprendizado *3D*, além da representação, é fornecido um *framework* para reconhecimento de objetos. São propostos dois algoritmos: algoritmo de *downsampling* e algoritmo de segmentação.

Um estudo sobre representações tridimensionais e o reconhecimento de padrões é realizado. A partir dos dados do acelerômetro do *wiimote* é proposto um método para reconhecimento de movimentos.

1.3 Estrutura do Trabalho

No Capítulo 2 são apresentados os algoritmos, as técnicas e o embasamento teórico sobre representação de conhecimento que serão utilizados ao longo do trabalho. No Capítulo 3 são apresentados conceitos de representações tridimensionais, dando enfoque às técnicas de mapeamento de dados tridimensionais: visão estereoscópica na Seção 3.1 e por meio de câmeras *RGB-D*, mais especificamente o *Kinect* na Seção 3.2. No Capítulo 4 é apresentado um estudo de caso do reconhecimento de padrões em estruturas tridimensionais com os dados capturados do *wiimote* e também de imagens capturadas do *Kinect*. Por fim, no Capítulo 5 são feitas as considerações finais.

2 *Materiais e métodos*

Este capítulo apresenta uma visão geral da teoria, algoritmos e técnicas que serão abordados ao longo do trabalho. Na Seção 2.1 é apresentado uma investigação dos formalismos de representação de conhecimento. Na Seção 2.1.1 é apresentado um estudo sobre as principais representações lógicas: lógica de primeira ordem, lógica proposicional e representação orientada a objetos. Na Seção 2.1.2 é discutido sobre a contrapartida existente entre expressividade e tratabilidade. Nas seções posteriores são abordadas as seguintes técnicas e algoritmos: KNN (Seção 2.2), TRKNN (Seção 2.3), DTW (Seção 2.4) e Algoritmos Genéticos (Seção 2.5).

2.1 Representação de Conhecimento

Nesta dissertação, como já mencionado, objetiva fugir da representação de conhecimento de tabelas atributo valor por meio de representações alternativas e mais ricas para as instâncias e exemplos. O foco deste trabalho está em representações tridimensionais dos dados, mais especificamente imagens tridimensionais e dados obtidos do sensor acelerômetro. Nesta seção é apresentado um estudo sobre as principais representações de conhecimentos conhecidas atualmente.

Discussões sobre conhecimento e sua representação datam desde a época dos grandes filósofos gregos. Entretanto, a primeira formalização real de representação de conhecimento surgiu quando lógicos matemáticos desenvolveram a arte de formalizar o conhecimento declarativo. Estavam primariamente interessados na formalização matemática.

A utilização deste tipo de representação apresentava-se limitada para a área de Inteligência Artificial, visto que o conhecimento não matemático exerce um importante papel. Esta ênfase era muito restrita da perspectiva de representação de conhecimento. Sua linguagem formal não era suficientemente expressiva. Mesmo com limitações, a lógica matemática clássica exerceu e têm exercido uma grande influência nas pesquisas de representações de conhecimento (PORTER, 2008).

A linguagem de fórmulas de primeira ordem é a linguagem da lógica clássica mais utilizada em teoria de representação de conhecimento. Proposta inicialmente para utilizar como um conhecimento declarativo por Hempeî (1965), posteriormente Robinson (1965) propôs prova automática por meio de raciocínio.

Segundo Porter (2008), lógica proposicional é o mais importante subconjunto de lógica de primeira ordem: o grande interesse se deu pelo fato da representação de conhecimento por fórmulas proposicionais estar relacionado a criação de resolvedores rápidos para a lógica proposicional.

2.1.1 Representações Lógicas

Linguagem por si só, representa um conjunto de palavras ou um conjunto de símbolos de algum tipo. Além propriamente da linguagem, segundo Brachman e Levesque (2004) é necessário considerar três aspectos:

1. sintaxe: especifica quais grupos de símbolos, organizados de uma forma específica são considerados formalmente corretos, ou seja, determina o formato em que os símbolos podem estar organizados;

2. semântica: especifica o que as expressões bem formadas devem significar. Para sentenças, é necessário a ideia ser clara sobre o mundo que está sendo expressado; e

3. pragmática: especifica como o significado das expressões na linguagem são utilizadas. Para representação de conhecimento, isto envolve em como são utilizadas as frases significativas de uma linguagem de representação, como parte de uma base de conhecimento a partir da qual inferências serão construídas.

Para Davis, Shrobe e Szolovits (1993), a noção de representação de conhecimento pode ser compreendida em termos de cinco papéis distintos que podem serem assumidos, cada um sendo importante para um tarefa específica:

1. Uma representação de conhecimento deve fundamentalmente ser um substituto para algo, permitindo assim que a entidade determine as consequências por meio do pensamento e não da ação, ou seja, racicionando sobre o ambiente ao invés de realizar ações sobre ele;

2. É um conjunto de compromissos ontológicos, isto é, uma resposta para uma pergunta;

3. É uma teoria fragmentada do raciocínio inteligente expressa em termos de três componentes principais:

    - a representação do conceito fundamental de raciocínio inteligente;
    - um conjunto de inferências que a representação sanciona;
    - um conjunto de inferências recomendadas;

4. É um meio computacional pragmaticamente eficiente, isto é, um ambiente computacional onde o pensamento pode ser realizado. Um fundamento para este argumento é provido pelo fato de a representação prover uma forma organizada de informação, facilitando desse modo o processo de inferir;

5. É um meio de expressão humana, isto é, uma linguagem em que é apresentado algo a respeito do mundo;

Posteriormente, são apresentadas três das principais representações lógicas encontradas na literatura: lógica de primeira ordem (FOL, *First-Order Logic*), lógica proposicional e representação orientada a objetos.

Lógica de Primeira Ordem

Criada pelo filósofo Gottlob Frege no final do século XX, o propósito inicial desta linguagem era para formalização da inferência matemática. Porém também foi adotada pela comunidade de Inteligência Artificial para propósitos de representação de conhecimento. Lógica de primeira ordem é suficiente para expressar uma boa quantidade de conhecimentos do senso comum (RUSSELL; NORVIG, 2009).

Dentro da linguagem FOL existem dois tipos de símbolos: *lógicos* e *não lógicos*. Intuitivamente, os símbolos lógicos são aqueles que tem um significado ou uso fixo na linguagem. Podem ser vistos sob três categorias:

1. pontuação: “)”, “(“ e “.”;

2. conectivos: “$\neg$”, “$\wedge$”, “$\vee$”, “$\exists$”, “$\forall$” e “$=$”. A interpretação destes símbolos é a mesma utilizada na matemática: “$\neg$” é negação, “$\wedge$” é conjunção lógica (“*e*”), “$\vee$” é disjunção lógica (“*ou*”), $\exists$ significa “existe ...”, $\forall$ significa “para todo ...” e “$=$” é igualdade lógica. “$\forall$” e “$\exists$” são chamados quantificadores;

3. variáveis: são utilizados como espaços reservados para a quantificação sobre objetos;

Quanto aos símbolos não lógicos, existem dois tipos:

- *símbolos função*: cada símbolo tem uma aridade (ou seja, um número específico de argumentos de entrada) e será interpretado como uma função mapeando os objetos de entrada para outro(s) objeto(s). Por exemplo, *PaiDe* é um símbolo função de aridade um. A interpretação natural é que *PaiDe(Francisco)* irá retornar o pai de Francisco. Símbolos funções de aridade zero são considerados constantes;

- *símbolos predicados*: a diferença entre símbolo função e símbolo predicado é semântica. Enquanto um símbolo função retorna um objeto, um símbolo predicado retorna o conceito de "verdadeiro" ou "falso". Como exemplo, considere o símbolo predicado de aridade um *Rico*. Então, *Rico(Francisco)* irá ser utilizado para denotar que *Francisco* é rico ou não. Símbolos predicados de aridade múltipla denotam relações entre objetos. Por exemplo, seja *Possui* um símbolo predicado de aridade dois, então *Possui(Francisco, Carro)* indica se Francisco possui um Carro.

Os símbolos não lógicos são aqueles em que o significado são dependentes da aplicação ou uso. Uma analogia com a linguagem da FOL e uma linguagem de programação seria que os símbolos lógicos são as palavras reservadas da linguagem de programação, enquanto que os símbolos não lógicos seriam como as variáveis e funções. É importante ressaltar que cada símbolo não lógico possui uma "aridade", ou seja, um número (não negativo) indicando quantos argumentos este símbolo utiliza.

Dois tipos de expressão sintática em FOL são permitidos: *termos* e *fórmulas*. Um termo se refere a algo no mundo, enquanto que uma fórmula é utilizada para expressar uma proposição. O conjunto de termos de FOL deverá satisfazer as mínimas condições:

- toda variável é um termo;

- se $t_1, ..., t_n$ são termos, e f é uma função símbolo de aridade n, então $f(t_1, ..., t_n)$ é um termo;

Um conjunto de fórmulas de FOL deve satisfazer as seguintes restrições:

- se $t_1, ..., t_n$ são termos, e P é um símbolo predicado de aridade n, então $P(t_1, ..., t_n)$ é uma fórmula;

- se t_1 e t_2 são termos, então $t_1 = t_2$ é uma fórmula;

- se α e β são fórmulas, e x é uma variável, então $\neg\alpha$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $\forall x.\alpha$, e $\exists x.\alpha$ são fórmulas;

Fórmulas dos dois primeiros tipos são denominadas *fórmulas atômicas*. Uma sentença em FOL é qualquer fórmula sem variáveis livres. A ocorrência de uma variável é dita limitada na fórmula se no escopo existir algum quantificador, e livre caso contrário. Isto é, x é limitada se aparece em uma subfórmula do tipo $\forall x.\alpha$ ou $\exists x.\alpha$ da fórmula. Por exemplo, dada a fórmula:

$$\forall y.P(x) \wedge \exists x[P(y) \vee Q(x)], \tag{2.1}$$

a primeira ocorrência da variável x está livre, ou seja, pode assumir qualquer valor. Enquanto que as duas ocorrências finais da variável x estão limitadas; ambas as ocorrências de y também estão limitadas. As sentenças de FOL constituem de fato a representação do conhecimento.

Exemplo

Suponha, conforme apresentado na Figura 2.1, a existência de três blocos coloridos sob uma mesa, onde: o bloco no topo possui cor roxa, o bloco no meio possui cor desconhecida e o bloco mais abaixo sabe-se a priori que não é roxo. O problema consiste em verificar se existe um bloco roxo no topo diretamente acima de um bloco não roxo.

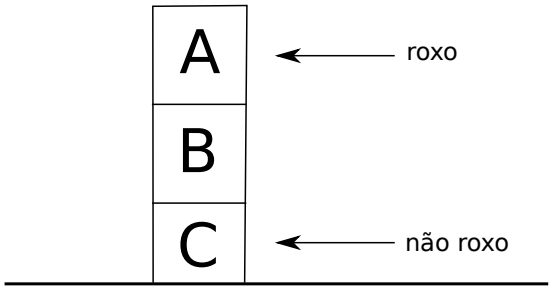


Figura 2.1: Uma pilha de três blocos (BRACHMAN; LEVESQUE, 2004)

A formalização do problema em *FOL* pode ser feita da seguinte forma: sejam a, b e c blocos e *Roxo* e *Acima* símbolos predicados indicando respectivamente, se um bloco é roxo e se um bloco está acima de outro. A partir da descrição, segue que:

$$Acima(a,b), Acima(b,c), Roxo(a), \neg Roxo(c).$$

O problema formalizado é suficiente para sua resolução. Os quatro fatos implicam que

existe, sem dúvida, um bloco roxo no topo de um bloco não roxo. Dessa forma, $S \models \alpha$ (ou seja, α é uma lógica consequência de S), onde α é:

$$\exists x \exists y. Roxo(x) \wedge \neg Roxo(y) \wedge Acima(x,y).$$

Para provar, deve-se demonstrar que qualquer interpretação que satisfaz S também satisfaz α. Satisfabilidade consiste em dado uma fórmula F e uma interpretação I, existe solução tal que $I \models F$?

Seja $\Im$ qualquer interpretação, e assuma que $\Im \models S$. Dois casos devem ser considerados:

1. $\Im \models Roxo(b)$. Como $\neg Roxo(c)$ e $Acima(b,c)$ estão em S, então

   $$\Im \models Roxo(b) \wedge \neg Roxo(c) \wedge Acima(b,c).$$

   Segue então que

   $$\Im \models \exists x \exists y. Roxo(x) \wedge \neg Roxo(y) \wedge Acima(x,y).$$

2. Suponha por hipótese que não seja verdade que $\Im \models Roxo(b)$. Então neste caso que $\Im \models \neg Roxo(b)$:

   $$\Im \models Roxo(a) \wedge \neg Roxo(b) \wedge Acima(a,b).$$

   Segue então que:

   $$\Im \models \exists x \exists y. Roxo(x) \wedge \neg Roxo(y) \wedge Acima(x,y).$$

De qualquer modo, $\Im \models \alpha$. Assim, α é logicamente consequência de S.

Uma linguagem de programação bastante conhecida que utiliza os conceitos de representações proposicionais e de lógica de primeira ordem como conceito central é *Prolog* (abreviação de *"PROgrammation en LOGique"*). Esta é uma linguagem de programação lógica de propósito geral associado com inteligência artificial e linguística computacional, sendo seu foco primário em lógica de primeira ordem. Foi concebida por um grupo liderado por Alain Colmerauer em Marseille, França, no início dos anos 70, sendo que o primeiro sistema desenvolvido foi em 1972 por Colmerauer com Philippe Roussel (COLMERAUER; ROUSSEL, 1996)

(KOWALSKI, 1988). Enquanto que inicialmente seu foco era processamento de linguagem natural, a linguagem tem desde então sido utilizada em uma grande gama de outras áreas como: provas de teoremas (STICKEL, 1988), sistemas especialistas (MERRITT, 1989), jogos, sistemas automáticos de resposta, ontologias e sistemas de controle sofisticados.

Lógica Proposicional

Esta seção é baseada no trabalho de (KLEMENT, 2005).

Lógica proposicional é um ramo da lógica no qual são estudados os vários modos de se unir e/ou modificar inteiras proposições, expressões ou sentenças para construir proposições, expressões ou sentenças mais complexas, assim como relações lógicas e propriedades derivadas destes métodos de combinar ou alterar expressões.

Na lógica proposicional, as expressões mais simples são consideradas como unidades indivisíveis, e por isso, na lógica proposicional não se estuda as propriedades lógicas e relações que dependem de partes de expressões que não expressões propriamente dita, tais como sujeito e predicados da expressão.

Uma expressão pode ser definida como uma sentença declarativa, ou parte de uma sentença, capaz de ser julgada a sua veracidade (verdadeiro ou falso). Abaixo seguem exemplos de expressões:

- *George W. Bush é o 43º Presidente dos Estados Unidos da América.*
- *Paris é a capital da França.*
- *Todo mundo que nasce na segunda-feira tem cabelo roxo.*

Algumas vezes, uma expressão pode ser composta de uma ou mais expressões. Considere, por exemplo, a seguinte expressão:

- *Ou Ganymede é lua de Jupiter ou Ganymede é lua de Saturno.*

Enquanto a sentença acima é uma expressão, porque é verdadeira, as duas partes *"Ganymede é a lua de Jupiter"* ou *"Ganymede é a lua de Saturno"*, são expressões, porque o primeira é verdadeira e a segunda é falsa.

Unir duas proposições simples com a palavra "e" é um modo comum de combinar expressões. Quando unidas, o complexo de expressões formado é verdadeiro se e somente se ambos

componentes são verdadeiros. Desse modo, um argumento da seguinte forma é logicamente válido:

- *Paris é a capital da França e Paris tem uma população de mais de dois milhões de habitantes.*
- *Então, Paris tem uma população de mais de dois milhões de habitantes.*

A lógica proposicional envolve o estudo de conectivos lógicos tais como as palavras "e" ou "ou" e regras determinando a validade das proposições. Além disso, estuda-se a modificação de expressões, como utilizar a palavra "não" para transformar uma expressão afirmativa para uma expressão negativa ou vice-versa. O princípio fundamental envolvido consiste em dado uma expressão verdadeira, a sua negação é uma expressão falsa, e dado que uma expressão é falsa, a sua negação é uma expressão verdadeira.

O diferencial entre lógica proposicional e outros ramos (tipicamente mais complicados) da lógica é que a primeira não lida com relações lógicas e propriedades que envolvem partes de uma expressão menores que simples expressões. Portanto, na lógica proposicional não se as características lógicas das proposições abaixo. Por exemplo:

- *George W. Bush é um presidente dos Estados Unidos da América.*
- *George W. Bush é um filho do presidente dos Estados Unidos da América.*
- *Então, existe alguém que é ambos: o presidente dos Estados Unidos da América e filho do presidente dos Estados Unidos da América.*

O reconhecimento de tal argumento requer que seja identificado que o *sujeito* da primeira premissa é o mesmo *sujeito* da segunda. Entretanto, na lógica proposicional, expressões simples são consideradas como indivisíveis, e então as relações lógicas e propriedades que envolvem as partes das expressões tais como *sujeitos* e *predicados* não são levados em consideração.

Primariamente lógica proposicional é o estudo de operadores lógicos. Um *operador lógico* é qualquer palavra ou frase que modifica uma expressão para torná-la uma expressão diferente ou uma junção de múltiplas expressões tornando-a mais complexa. Em inglês, palavras tais como *"and"*, *"or"*, *"not"*, *"if .. then"*, *"because"* e *"necessarily"* constituem o conjunto de operadores.

Embora não trivial, a lógica proposicional está implicitamente presente na grande maioria das áreas de aprendizado de máquina. Como entrada, a maioria dos algoritmos indutores em

aprendizado de máquina correspondem a uma tabela atributo valor, conforme apresentado na Tabela 2.1

Tabela 2.1: Exemplo de uma tabela atributo valor (MITCHELL, 1997a)

Exemplo	Aparência	Temperatura	Umidade	Ventando	Viajar
E_1	sol	25	72	sim	bom
E_2	sol	28	91	sim	ruim
E_3	sol	22	70	não	bom
E_4	sol	23	95	não	ruim
E_5	sol	30	85	não	ruim
E_6	nublado	23	90	sim	bom
E_7	nublado	29	78	não	bom
E_8	nublado	19	65	sim	ruim
E_9	nublado	26	75	não	bom
E_{10}	nublado	20	87	sim	bom
E_{11}	chuva	22	95	não	bom
E_{12}	chuva	19	70	sim	ruim
E_{13}	chuva	23	80	sim	ruim
E_{14}	chuva	25	81	não	bom
E_{15}	chuva	21	80	não	bom

Esta tabela atributo valor pode ser representado utilizando lógica proposicional:

$$(Aparencia = sol) \wedge Temperatura = 25 \wedge Umidade = 72 \wedge Ventando = sim \wedge Viajar = bom) \vee$$
$$(Aparencia = sol) \wedge Temperatura = 28 \wedge Umidade = 91 \wedge Ventando = sim \wedge Viajar = ruim) \vee$$

...

Diversos classificadores gerados por algoritmos indutores na área de aprendizado de máquina (por exemplo, árvores de decisão, regras, etc) podem ser representados em termos de lógica proposicional. Suponha, por exemplo, a árvore apresentada na Figura 2.2.

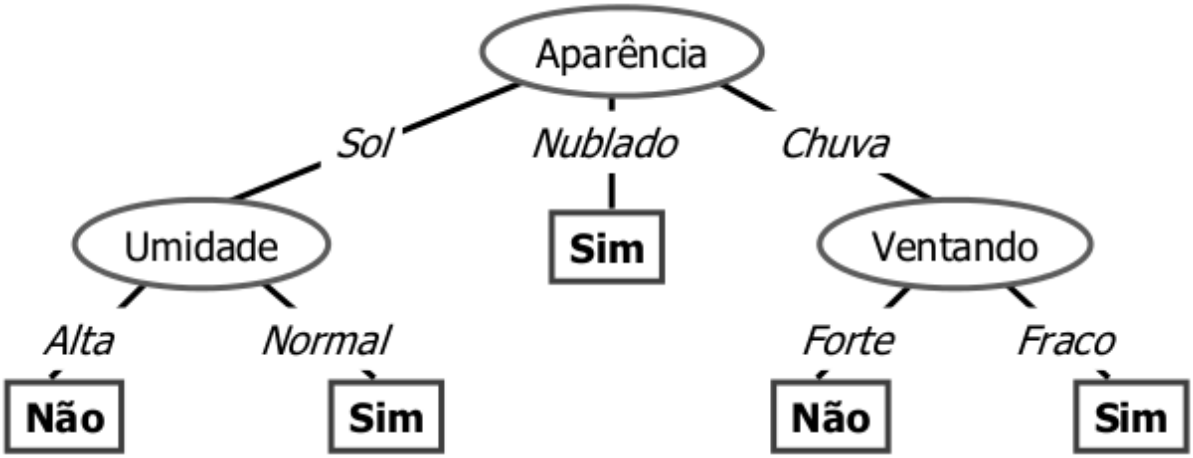


Figura 2.2: Exemplo de uma árvore de decisão (MITCHELL, 1997a)

Um caminho na árvore é composto pelos nós (comparações) da raiz até uma folha qualquer. Cada caminho é independente, ou seja, os exemplos cobertos por dois caminhos quaisquer são sempre disjuntos. A união (conjunção) de todos os caminhos compõe a árvore. Portanto, pode-se visualizar a árvore como conjunções de disjunções. Outra maneira de representar um caminho na árvore seria por uma regra, e consequentemente lógica proposicional. A árvore seria o conjunto de conjunções de regras. Segue abaixo:

SIM

$$(Aparencia = Sol \wedge Umidade = Normal) \vee$$

$$(Aparencia = Nublado) \vee$$

$$(Aparencia = Chuva \wedge Ventando = Fraco)$$

NÃO

$$(Aparencia = Sol \wedge Umidade = Alta) \vee$$

$$(Aparencia = Chuva \wedge Ventando = Forte)$$

Representação Orientada a Objetos

Por diversas vezes em uma base de conhecimento, o conhecimento sobre um determinado objeto ou tipo de objeto pode estar espalhado por toda a base de conhecimento. De acordo com o crescimento do número de sentenças ou procedimentos em uma base de conhecimento,

a organização deste conhecimento é algo crítico. Uma abordagem de representação bastante conhecida consiste no agrupamento de fatos ou regras em termos de tipos de objetos à que eles pertencem. Sem dúvida, é muito natural pensar em conhecimento não como uma mera coleção de sentenças, mas sim como uma forma estruturada e organizada em termos do que o conhecimento é sobre. Esta forma de organização é chamada de objetos do conhecimentos (BRACHMAN; LEVESQUE, 2004).

Objetos podem representar tanto algo físico, tal como uma casa ou uma pessoa, quanto algo conceitual, tal como um curso ou uma viagem, ou até mesmo uma abstração, tal como um evento ou uma relação. Cada um destes tipos de objetos podem possuir as suas próprias partes, algumas físicas (telhado, porta, quarto, etc.; braço, cabeça, perna, etc.), e outras abstratas (nome do curso, professor, estudante, horário do encontro, etc.). Entre as partes, existem diversas restrições: o telhado deve estar conectado às paredes de uma maneira, toda pessoa deve ter um nome, etc. As restrições entre as partes devem estar expressas de forma procedural, tais como: procedimento de registro que conecta um estudante à um curso, ou um procedimento para reservar um assento de avião que conecta um segundo destino de uma viagem. Também, alguns tipos de objetos podem ter procedimentos de algum outro tipo que são cruciais para a nossa compreensão: procedimento para reconhecer banheiros em casas, para reservar quartos de hotéis em viagens entre outros. Em geral, em um sistema de representação orientada a objetos procedural, deve-se considerar tipos de operações de raciocínio que são relevantes para os vários tipos de objetos na aplicação.

Proposta em 1975 por Marvin Minsky, a ideia de utilizar grupos orientados à objetos de procedimentos para reconhecer e lidar com novas situações foi inicialmente pensada como uma representação de conhecimento com foco em reconhecimento. Apesar disso, a ideia de agrupar procedimentos relacionados desta maneira para raciocinar teve uma aplicação muito maior. É importante notar que em sua publicação, Marvin utiliza-se do termo quadro ao invés de objetos para se referir a estrutura de dados proposta.

Existem dois tipos básicos de quadros: quadros individuais, utilizado para representar objetos únicos e quadros genéricos, utilizado para representar categorias ou classes de objetos. Um quadro individual consiste em uma lista nomeada de locais, onde valores podem ser dispostos. Os locais são chamados *slots*, e os itens que podem ser dispostos dentro dele são denominados *fillers*. Esquematicamente, um quadro individual possui o seguinte formato:

(*Nome do quadro*
<*nome-slot1 filler1*>
<*nome-slot2 filler2*>

...)

Os nomes do quadro e do *slot* devem ser símbolos atômicos; no *filler* pode ocorrer tanto valores atômicos (como números ou *strings*) ou nomes de outros quadros individuais. Abaixo segue um exemplo de representação utilizando um quadro individual:

(tripLeg123
<:INSTANCE-OF TripLeg>
<:Destino toronto> ...)

(toronto
<:**INSTANCE-OF** CidadeCanadense>
<:Provincia ontario>
<:População 4.5M> ...)

Quadro individual também possui uma característica especial: um *slot* chamado :**INSTANCE-OF**, no qual o *filler* é o nome de um quadro genérico indicando a categoria do objeto sendo representado. Diz-se que o quadro individual é um instância de um quadro genérico, neste caso, *toronto* é uma instância de *CidadeCanadense*.

Um conceito central em representação orientada à objetos é "herança". Muito do que acontece da atividade de raciocínio que é feito com o sistema de quadros envolve a criação de instâncias individuais de quadros genéricos, preenchendo alguns *slots* com valores e inferindo alguns outros valores. A representação deste conceito pode ser feita pelos itens **:INSTANCE-OF** e **:IS-A**. Em particular, os quadros genéricos podem ser utilizados para preencher valores que não são mencionados explicitamente na criação da instância, e eles podem ativar ações adicionais onde o preenchimento de *slots* são providos.

2.1.2 Expressividade e Tratabilidade

O que se deseja é que a representação seja capaz de representar formalmente qualquer coisa. Por que não então utilizar uma linguagem altamente expressiva? A primeira sugestão para essa questão poderia ser definir uma linguagem de representação de conhecimento formal similar a linguagens como *Português* ou *Inglês*. Porém, é preciso estar atento a dois conceitos: representação e raciocínio. Do ponto de vista da representação, uma linguagem altamente expressiva e rica é, sem dúvida, uma linguagem que atende de forma excelente a capacidade de representação. Enquanto isso, esta mesma linguagem pode ser vista como "pobre" do ponto de vista do raciocínio. Atualmente, procedimentos de raciocínio que são requeridos para tratar

com uma linguagem de representação mais expressiva parecem não funcionar na prática. Um fato fundamental dentro de representação de conhecimento é que existe um compromisso entre a expressividade e a tratabilidade das tarefas de raciocínio.

A partir da Seção 2.2 são abordadas as técnicas e métodos que serão utilizados ao longo deste trabalho.

2.2 *k-Nearest Neighbors*

k-Nearest Neighbors (k-vizinhos mais próximos em português) (MITCHELL, 1997b), mais conhecido como KNN é um método para classificação de objetos baseados nos exemplos mais próximos em um espaço de características. Os exemplos de treinamento são representados em um espaço *n*-dimensional, onde *n* é o número de atributos, e cada exemplo de treinamento é representado neste espaço. KNN é um tipo de aprendizado por instância, ou algoritmo preguiçoso onde a função é apenas aproximada localmente e todo o processamento é postergado até o momento da classificação. O KNN está entre os algoritmos de aprendizado de máquina mais simples: um objeto é classificado pelo voto majoritário dos seus vizinhos, com todos os objetos sendo atribuídos para a classe mais comum entre os k vizinhos mais próximos (k é um inteiro positivo, tipicamente pequeno). Se k = 1, então o objeto é simplesmente atribuído para a classe do vizinho mais próximos.

Normalmente tarefas de reconhecimento de movimento requerem o reconhecimento de diversos movimentos por meio de um único classificador. KNN é naturalmente multiclasse não necessitando de adaptações no método. Outra característica está no tempo de treinamento. O treinamento do algoritmo consiste no armazenamento dos exemplos de treinamento, pertencendo na literatura a família de algoritmos *lazy*. A classificação de um novo exemplo consiste na verificação da classe dos *k* vizinhos mais próximos. A classe que estiver com maior proporção de vizinhos é utilizada para rotular o novo exemplo. O funcionamento básico do algoritmo KNN é apresentado na Figura 2.3. Juntos, quadrados e triângulos formam o conjunto de treinamento. Dado um novo exemplo (círculo), o algoritmo classifica este exemplo a partir do cálculo da classe majoritária. A classificação consiste no cálculo da distância entre o novo exemplo e todos os exemplos armazenados. Considera-se as *k* menores distâncias. A classe que possuir mais exemplos dentre os *k* exemplos selecionados, corresponde a classe do novo exemplo, no caso da Figura o novo exemplo será classificado como sendo da classe triângulo.

O KNN utilizado neste trabalho é uma adaptação da sua forma original, para que se dê um peso maior aos elementos mais próximos da nova entrada. O método pode ser definido da

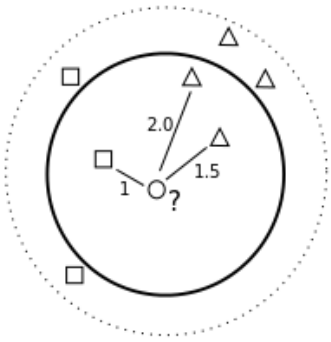


Figura 2.3: Objetos podem ser classificados em duas categorias: triângulos ou quadrados. O elemento círculo representa uma nova entrada que ainda não foi classificada

seguinte forma: seja x um novo exemplo. Deve-se calcular a distância d de x a todos os outros pontos existentes (conjunto de treinamento). Seleciona-se as k menores distâncias e aplica-se a Fórmula 2.2, atribuindo-se dessa forma um peso maior aos elementos da classe que estiverem mais próximos.

Seja C o conjunto de todas classes dos k primeiros elementos. Para cada classe $c \in C$, calcula-se:

$$w_c = \sum_{j=1}^{i} \frac{1}{d^2} \tag{2.2}$$

onde i é o número de exemplos pertencentes a classe c e d é a distância do exemplo i ao novo exemplo que deve ser classificado. O novo exemplo pertencerá a classe que obtiver o maior valor de w_c.

Uma característica indesejável de KNN está associado ao tempo de classificação de novos exemplos, especialmente quando o conjunto de treinamento utilizado for muito grande. Seja m o número de exemplos de treinamento, n o número de atributos e $k > 1$. Para classificar um novo exemplo, é necessário calcular a distância desse novo ponto à todos os m pontos de treinamento, o que levaria $Acima(nm)$ passos para cada exemplo de teste. A classificação de um único exemplo requer o acesso de todos os valores da matriz que representa o conjunto de treinamento tornando o uso deste algoritmo proibitivo para problemas que requerem rápida resposta.

Atento a esta limitação, este trabalho utiliza-se de uma técnica para redução do número de exemplos de treinamento (pontos). A redução é obtida pelo descarte dos pontos do treinamento que estão afastados das bordas de decisão ou seja aqueles que possuem uma pequena influência na classificação pelo KNN. Esta técnica é proposta por (FAYED; ATIYA, 2009), e é denominada *Template Reduction For* KNN (TRKNN).

2.3 TRKNN - *Template Reduction for k-Nearest Neighbors*

Aparentemente no algoritmo KNN, os exemplos com mais informação são os exemplos que estão no centro das representações das classes no plano *n* dimensional. Intuitivamente estes centros representam a generalização da classe. Assim, na tentativa de reduzir a quantidade de exemplos de treinamento, é natural em um primeiro momento, pensar em manter os exemplos que representam os centróides das classes e remover as bordas. Neste pensamento incorre um grande perigo: a destruição da borda de decisão.

Imagine que duas classes, positivo e negativo, sejam representados como dois retângulos iguais, um imediatamente ao lado do outro em um espaço bidimensional. Ao manter apenas os centróides das classes e remover os exemplos restantes, o conceito representado são duas circunferências, uma para cada classe, perdendo-se a noção de classes retangulares. Ao realizar o inverso, manter os exemplos da borda de decisão e remover os exemplos do centro do retângulo, o conceito de retângulo continua sendo representado pelos exemplos.

Ao remover os exemplos das bordas de decisão ocorre um aumento no *bias* do algoritmo reduzindo o poder de KNN de representar hipóteses mais complexas. Mantendo apenas os centróides, os conceitos que podem ser aprendidos são apenas aqueles com formato esférico ou circular. Em outras palavras, mantendo os centróides é possivel apenas aprender conceitos no raio de distância dos centróides. Assim, manter os exemplos da borda possibilita o KNN representar hipóteses mais complexas.

Pensando neste problema, (FAYED; ATIYA, 2009) mantém os exemplos da borda utilizando uma cadeia denominada *Nearest Neighbor Chain* (Cadeia de Vizinhos Mais Próximos) que consiste de uma sequência de vizinhos mais próximos de classes alternantes. Para exemplificar considere um exemplo da classe positiva escolhida de maneira arbitrária.

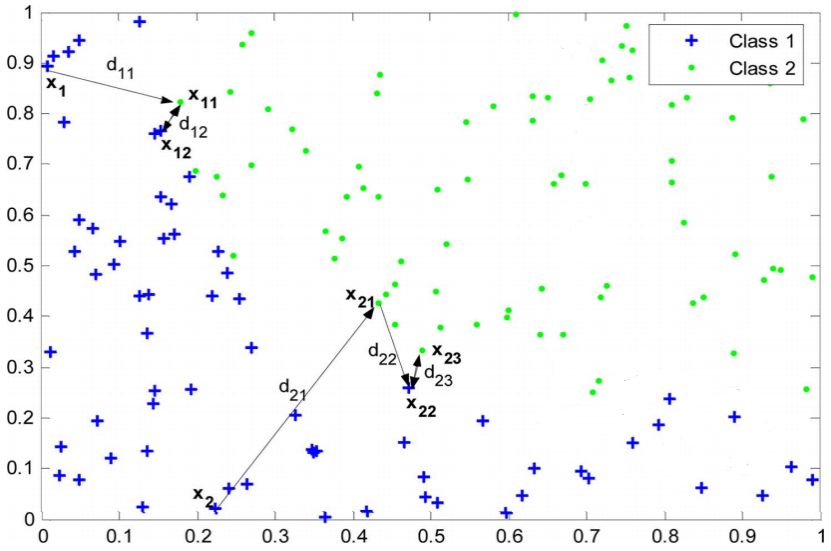


Figura 2.4: Exemplo da abordagem desenvolvida utilizando DTW + TRKNN. Figura alterada de (FAYED; ATIYA, 2009)

Na Figura 2.4 é apresentada uma visão geral do algoritmo. O algoritmo irá procurar pelo vizinho mais próximo deste exemplo positivo que seja da classe negativa. O exemplo da classe negativa irá procurar pelo vizinho mais próximo da classe positiva, ignorando os vizinhos já visitados. De maneira alternada, positivo e negativo, é formada uma cadeia de exemplos que compõem a borda de decisão entre positivo e negativo. A partir da construção desta cadeia, são descartados exemplos que estão longe das bordas, ou seja, possuem pouca influência na classificação do KNN. De maneira mais formal considere a seguinte definição:

Definição: Uma *Nearest Neighbor Chain* C_i de um padrão x_i (da classe ω_m) é definida pela sequência x_{i0}, x_{i1}, x_{i2} ... x_{ik} e pela sequência $d_{i,0}$, $d_{i,1}$ ... $d_{i,k-1}$, onde a raíz da cadeia $x_{i0} = x_i$, e x_{ij} é o vizinho mais próximo do padrão $x_{i,j-1}$ (de uma classe diferente de ω_m se j é ímpar e da classe ω_m se j é par). Além disso, $d_{ij} = ||x_{i,j+1} - x_{i,j}||$ é a distância euclidiana entre os padrões $x_{i,j}$ e $x_{i,j+1}$. A condição de parada na formação da cadeia é dada no elemento x_{ik} se $x_{i,k+1} = x_{i,k-1}$. É importante notar que a sequência de distâncias é uma sequência não crescente (ou seja, $d_{ij} > d_{i,j+1}$).

Uma vez que as cadeias estejam formadas, (FAYED; ATIYA, 2009) propõe um método de condensação. A ideia básica para o método de condensação é que para cada exemplo x_i contido no conjunto de treinamento, é construída uma cadeia C_i. O padrão x_{ij} na cadeia é eliminado (do conjunto condensado) se $d_{ij} > \alpha * d_{i,j+1}$, onde α é um limiar > 1 e $j = 0, 2, 4...$ até o tamanho da cadeia. Note que apenas padrões da mesma classe de x_i podem ser eliminados (ou seja, são considerados apenas os padrões pares na cadeia). Isto é importante quando se está lidando com problemas de multiclasse, uma vez que a cadeia é construída utilizando o conceito um contra

todos.

Utilizando-se desta abordagem, existe uma melhora no desempenho em termos de tempo de execução durante a classificação de novos exemplos em comparação com o KNN puro, uma vez que existe uma redução no número de elementos armazenados. A princípio, não deverá existir alteração na acurácia da classificação, pois TRKNN elimina apenas os elementos que estão fora do limite de classificação, ou seja, que possuem pouca ou nenhuma influência na classificação de novos elementos.

Ter um algoritmo de rápido treinamento e classificação de exemplos não é suficiente para solucionar o problema de reconhecimento de movimentos. O algoritmo KNN na sua forma original trabalha com exemplos no formato de tabelas atributo valor para calcular as distâncias entre os exemplos. Entretanto KNN não precisa que os dados estejam obrigatoriamente em um formato de tabela atributo-valor, mas precisa das distâncias entre os exemplos. Neste trabalho é utilizado o algoritmo DTW (KEOGH; RATANAMAHATANA, 2005a) para o cálculo de distâncias entre as séries temporais explicado a seguir.

2.4 DTW- *Dynamic Time Warping*

Em ciência, é bastante comum séries (padrões) que apesar da aparente similaridade, estão dispostas sob diferentes unidades, amplitudes ou tempos. Em reconhecimento de voz, por exemplo, uma mesma palavra pode ser dita de diversas maneiras: pode-se falar alto, baixo, rápido ou devagar, uma mesma palavra pode sofrer variações na amplitude e/ou no tempo. Mesmo com esta alteração, seres humanos ainda são capazes de reconhecer a pronúncia de uma palavra. Para comparar duas sequências temporais, muitos algoritmos utilizam a distância euclidiana (ou alguma variação). Entretanto, para o caso de séries temporais, a distância euclidiana induz a erros, pois assume que o i-ésimo ponto é alinhado com o i-ésimo ponto na outra sequência e pode produzir uma medida equivocada. A variabilidade do eixo do tempo pode causar casamentos errôneos quando se utiliza distância euclidiana.

O algoritmo DTW é capaz de medir a similaridade entre duas sequências mesmo que estejam deformadas em algum eixo. Em comparação com a distância euclidiana, DTW é uma medida de distância muito mais robusta, principalmente para o caso de séries temporais, uma vez que permite o casamento de formas similares que estão em diferentes fases (KEOGH; RATANAMAHATANA, 2005b). O aparecimento comum de dados na forma de séries temporais torna o DTW bastante versátil. DTW tem se mostrado eficaz em diversas situações, como por exemplo: casamento de impressões digitais (KOVACS-VAJNA, 2000), reconheci-

mento de escrita (MYERS; RABINER, 1981; RATH; MANMATHA, 2003; BAHLMANN; BURKHARDT, 2004), reconhecimento de voz (YU, 1995; RABINER; JUANG, 1993), medicina (CAIANI, 1998).

DTW pode ser descrito formalmente da seguinte forma: dadas duas séries X e Y, tal que $X = \{x_1, x_2, \ldots, x_i, \ldots, x_{|n|}\}$ e $Y = \{y_1, y_2, \ldots, y_i, \ldots, y_{|m|}\}$, para alinhar as duas sequências é necessário construir uma matriz tal que cada posição é calculada da seguinte maneira

$$D(i,j) = d(x_i, y_j) + \min \begin{cases} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{cases},$$

no qual $D(i,j)$ representa a distância entre a sequência $X[1..i]$ e $Y[1..j]$, contendo na posição $D(n,m)$ a distância entre as duas sequências inteiras. A partir desta técnica é possível medir a semelhança entre duas séries temporais e fornecer a distância entre elas. Neste trabalho, o algoritmo DTW é aplicado a duas imagens, no qual os *pixels* das imagens correspondem as sequências e a verosimilhança (distância) calculada através do DTW representa a semelhança entre as duas imagens, de forma que quanto menor a distância obtida pelo algoritmo, maior a semelhança entre as duas imagens.

Na Figura 2.5 são ilustrados dois alinhamentos de séries temporais, uma utilizando a distancia euclidiana e a outra utilizando DTW. Note que, embora as duas séries tenham uma forma global similar, elas não estão alinhadas ao eixo x. A distância euclidiana, que assume que o i-ésimo ponto em uma sequência é alinhado com o i-ésimo ponto da outra sequência, irá produzir uma medida de similaridade pessimista. A técnica DTW permite uma medida de distância mais intuitiva.

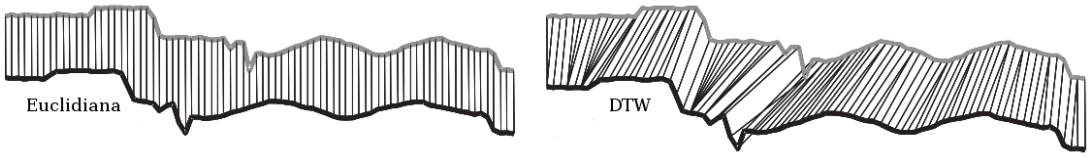


Figura 2.5: Comparação de séries temporais utilizando distância euclidiana e DTW (KEOGH; RATANAMAHATANA, 2005a)

Suponha a existência de duas séries temporais, uma sequência Q de tamanho n e uma sequência C de tamanho m como ilustrado na Figura 2.6, onde

$$Q = q_1, q_2, ..., q_i, ...q_n, \tag{2.3}$$

$$C = c_1, c_2, ..., c_i, ...c_m. \tag{2.4}$$

Para alinhar estas duas sequências utilizando DTW, primeiro deve-se construir uma matriz de tamanho $n \times m$ onde o elemento da matriz (i^{th}, j^{th}) corresponde a distância quadrada, $d(q_i, c_j) = (q_i - c_j)^2$, que é o alinhamento entre os pontos q_i e c_j. Para encontrar o melhor emparelhamento entre estas duas sequências, obtêm-se um caminho através da matriz que minimiza a distância acumulativa entre elas, como mostrado na Figura 2.5. Em particular, o caminho ótimo é o caminho que minimiza o custo *warping*

$$DTW(Q,C) = min\{\sqrt{\sum_{k=1}^{K} w_k}\}, \tag{2.5}$$

no qual w_k é o elemento da matriz $(i, j)_k$, que também pertence ao k^{th} elemento do caminho *warping* W, um conjunto de elementos contínuo da matriz que representa um mapeamento entre Q e C.

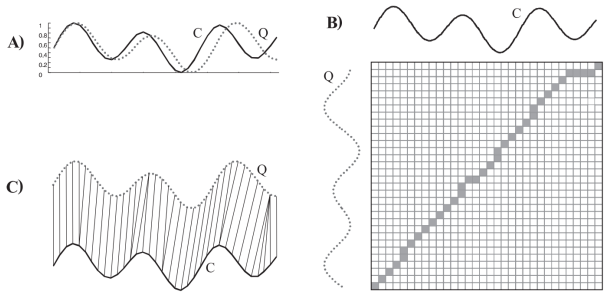


Figura 2.6: A - Sequências Q e C; B - Para o alinhamento de sequências é construído a matriz de *warping* onde o caminho ótimo é mostrado nos quadrados em negrito; C - Alinhamento resultante (KEOGH; RATANAMAHATANA, 2005a)

Este caminho *warping* pode ser encontrado utilizando programação dinâmica para avaliar a seguinte recorrência.

$$\gamma(i, j) = d(q_i, c_j) + min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}, \tag{2.6}$$

no qual $d(i, j)$ é a distância encontrada da célula corrente, e $\gamma(i, j)$ é a distância acumulativa de $d(i, j)$ e a menor distância acumulativa das três células adjacentes. Na Figura 2.6 B é apresentada a matriz formada pelo cálculo de $\gamma(i, j)$ formando a matriz de deformação para todas as combinações das sequências Q e C. Os quadrados em negrito que formam o uma linha na

diagonal da matriz representam o caminho ótimo a ser utilizado para a transformação de Q em C. A valor da posição $\gamma(n,m)$ da matriz de deformação é utilizada como medida de distância entre as duas sequências representando a quantidade de deformações (*warpings*) necessários para que a transformação ocorra.

2.5 Algoritmo Genético

Algoritmos genéticos são uma família de modelos computacionais inspirados na evolução (WHITLEY, 1994). Baseado no princípio proposto por Darwin de seleção natural, o algoritmo genético realiza uma busca pelo indivíduo mais apto dentro de uma população, sendo muito utilizado em técnicas de otimização e ajustes de parâmetros. A ideia central destes algoritmos está em codificar a solução potencial como um cromossomo, onde são aplicados operadores de recombinação e mutação. Para executar um algoritmo genético é necessário primeiramente uma codificação (representação) adequada. Além disso, é necessário uma função objetivo, que tem por objetivo atribuir um *rank* para cada solução codificada. Durante o processo de execução, os pais devem ser selecionados para reprodução e recombinação (BEASLEY, 1993). Sendo assim, os aspectos principais de um algoritmo genético são:

- Codificação: Uma solução potencial para o problema é representada como um conjunto de parâmetros. Estes parâmetros então são unidos para formarem um conjunto de valores;
- Função Objetivo: Permite avaliar quão bom uma solução potencial é. Para cada problema em particular deve existir uma função objetivo;
- Reprodução: Durante a fase reprodutiva de um algoritmo genético, indivíduos são selecionados de uma população e recombinados, gerando descendentes que irão compor a próxima geração. O processo de recombinação geralmente funcionada de duas formas: cruzamento, no qual são selecionados dois indivíduos e corta-se a sua cadeia de cromossomos em uma posição aleatória, quebrando-o em duas partes. Estas duas partes são então trocadas entre dois indivíduos e mutação, no qual são alterados pequenas quantidades de genes de forma randômica. Cruzamento é uma recombinação que exige dois indivíduos, enquanto que mutação é uma recombinação individual;
- Convergência: é o progresso da uniformidade da população. Um gene é tido que foi convergido quando 95% da população compartilha o mesmo valor. A população é dita que foi convergida quando todos os genes convergiram.

Várias são as abordagens existentes em aplicações utilizando algoritmos genéticos, de uma forma geral o algoritmo começa com cada indivíduo da população sendo inicializado com valores aleatórios, e a cada iteração do algoritmo, chamada geração, os indivíduos são avaliados e lhes é atribuído um objetivo, que representa o quão bom é o indivíduo. De acordo com o objetivo de cada indivíduo, uma nova população é gerada, sendo que os mais aptos permanecem intactos e os outros indivíduos da população passam por processos de cruzamento e mutação, gerando assim uma nova população.

Exemplo

Para uma melhor ideia do funcionamento do algoritmo genético, nesta seção é apresentado um exemplo. Dada uma função $f(x) = x^2$, o objetivo consiste em encontrar o valor x para o qual a função assume o valor máximo utilizando a técnica de algoritmo genético, onde $0 \leq x \leq 31$.

Claramente pode ser observado que o valor máximo é obtido quando $x = 31$. Entretanto, o importante não é o resultado em si, mas como alcançá-lo por meio da técnica de algoritmo genético. De maneira geral, um algoritmo genético é iniciado com uma população escolhida aleatoriamente, no qual cada indivíduo desta população corresponde a uma série binária.

A seleção é uma parte chave do algoritmo. Em geral, usa-se o algoritmo de seleção por "roleta" sendo os indivíduos ordenados de acordo com a função-objetivo e lhes são atribuídas probabilidades decrescentes de serem escolhidos - probabilidades essas proporcionais à razão entre a adequação do indivíduo e a soma das adequações de todos os indivíduos da população. A escolha é feita então aleatoriamente de acordo com essas probabilidades. Dessa forma consegue-se escolher como pais os mais bem adaptados, sem deixar de lado a diversidade dos menos adaptados. Outras formas de seleção podem, ainda, ser aplicadas dependendo do problema a ser tratado. Como exemplos pode-se citar a seleção por "torneio" (onde são selecionados diversos pequenos subconjuntos da população, sendo selecionado o indivíduo de maior adequação de cada um desses grupos), a seleção por "classificação" ou "ranking" (semelhante à seleção por "roleta", com a diferença de que a probabilidade de seleção é relacionada à sua posição na ordenação dos indivíduos da população e não à sua adequação em si) e a seleção por "truncamento" (onde são selecionados os N melhores indivíduos da população, descartando-se os outros).

No exemplo a seguir será utilizada a seleção por "roleta". A função objetivo determina a qualidade da série e baseado nesta informação, a probabilidade do indivíduo ser selecionado para o cruzamento para gerar os filhos (prole) da próxima geração. Então, é realizado *crossover* entre duas cópias selecionadas aleatoriamente e uma seleção, também aleatória, do tamanho da

cadeia onde será realizado o cruzamento ou troca (conforme presentado na Figura 2.7).

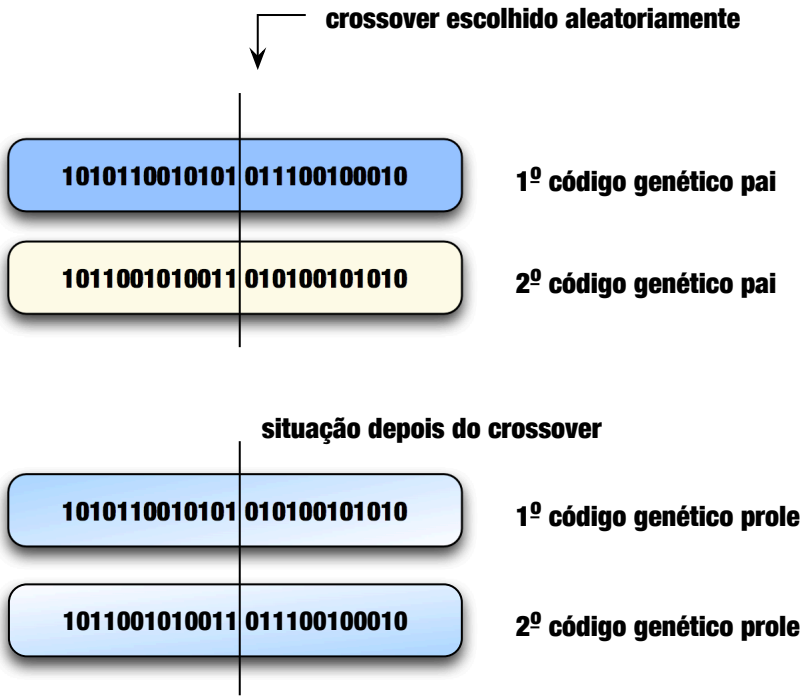


Figura 2.7: Código genético dos pais e da prole antes e depois do cruzamento

Na Figura 2.7 é apresentado o código genético dos pais e da prole antes e depois do cruzamento. Um bom indivíduo tem uma probabilidade muito maior de ser escolhido como pai em comparação com um indivíduo considerado ruim. Além do conceito de cruzamento, existe também o conceito de mutação, que neste experimento foi considerado com a probabilidade 0.001 de ocorrência. Isto significa que existe a probabilidade de um em mil para cada *bit* na série ser alterado de 0 para 1, ou o inverso. Neste experimento (Tabelas 2.2 e 2.3), nenhum *bit* foi mutado na primeira geração. A execução do algoritmo continua até ser atingido o indivíduo com qualidade $31^2 = 961$, atendendo o critério de parada, ou até o número de iterações atingir o valor limite fornecido a priori. No útimo caso, o algoritmo não encontrou a solução; ao menos não a solução ótima determinada no início.

A execução do algoritmo genético apresentado é simples, porém os conceitos são os mesmos para qualquer problema que se deseje resolver com algoritmo genético. Abaixo segue a ideia dos passos do algoritmo genético:

Conforme apresentado no Algoritmo 1, o algoritmo genético é iniciado com uma população (normalmente aleatória) e uma função objetivo. Os passos de selecionar a lista de pais e atualizar a população (realizar cruzamento e/ou mutação) é realizado enquanto nenhuma das condições for atingida: alcançar o valor da função objetivo desejada ou atingir o número limite

Algoritmo 1: Algoritmo Genético

Entrada: população, função-fitness
1 **enquanto** *nenhuma condição de parada for atingida* **faça**
2 lista de pais := seleção(população, função-fitness)
3 população := reprodução(lista de pais)
4 **retorna** *melhor indivíduo da população de acordo com a função-fitness*

Tabela 2.2: Computação da qualidade de uma população iniciada aleatoriamente

# da Série	População Inicial	x	$f(x) = x^2$ f_i	Objetivo Prole f_i/f_{avg}	Objetivo Prole f_i/f_{avg} (arredondado)
1	0 1 1 0 1	13	169	0.58	1
2	1 1 0 0 0	24	576	1.97	2
3	0 1 0 0 0	8	64	0.22	0
4	1 0 0 1 1	19	361	1.23	1
Soma			1170	4.00	4
Média: f_{avg}			239	1.00	1
Máximo			576	1.97	2

de iterações conhecida a priori.

A Tabela 2.2 apresenta um exemplo com as qualidades computadas de uma população iniciada aleatoriamente (primeira geração). Com base nas características computadas dos indivíduos, a seleção é feita e apenas os melhores indivíduos são permitidos continuar com o cruzamento (multiplicação). Na Tabela 2.3 é apresentado o cruzamento da primeira geração e o "nascimento" de uma prole e suas qualidades computada com a função objetivo. Quando é realizado o cruzamento entre os pais (primeira coluna da Tabela 2.3), obtêm-se a população com características melhores (como pode ser observado na Tabela 2.3 - colunas 4-6). É assumido o valor 0.001 para a mutação. Uma vez que existem quatros indivíduos em cada ge-

Tabela 2.3: Apresentação do cruzamento e computação das qualidades da prole da primeira geração

Primeira Geração	Parceiro Cruzamento (aleatório)	Ponto de Cruzamento (aleatório)	Nova População	x	$f(x) = x^2$ f_i/f_{avg} (arredondado)
0 1 1 0 | 1	2	4	0 1 1 0 0	12	144 (0.32, 0)
1 1 0 0 | 0	1	4	1 1 0 0 1	25	625 (1.42, 1)
1 1 | 0 0 0	4	2	1 1 0 1 1	27	729 (1.66, 2)
1 0 | 0 1 1	3	2	1 0 0 0 0	16	256 (0.58, 1)
Soma					1754
Média: f_{avg}					439
Máximo					729

ração, cada indivíduo de tamanho cinco *bits*, a probabilidade de ocorrer mutação em um deles é 4*5*0.001=0.02. No primeiro passo, não ocorreu mutação em nenhum dos *bits*. A computação dos valores é continuada até o critério de otimização (x = 31) ou número de iterações limite ser atingido.

Algumas modificações e melhorias devem ser realizadas para aplicar a técnica de algoritmo genético na solução de problemas mais complexos, entretanto este assunto está além desta seção.

3 *Mapeamento de Representações Tridimensionais*

Atualmente, existem diversas maneiras para aquisição de imagens com profundidade (x, y, z). Entre as mais conhecidas destacam-se: escaneamento laser (VOSSELMAN, 2004), visão estereoscópica (KLETTE, 1998) e mais recentemente RGB-D (notadamente *Kinect*) (HENRY, 2010). O conceito central na representação tridimensional consiste em utilizar uma maior quantidade de informação quando se comparada a representações bidimensionais. Neste capítulo é apresentado duas abordagens particulares de mapeamento de imagens tridimesionais: visão estereoscópica (Seção 3.1) e câmeras *RGB-D* (Seção 3.2).

Em uma imagem obtida a partir de uma câmera comum, a unidade básica de informação é o *pixel* (menor unidade de uma imagem que pode ser representada ou controlada), onde neste caso armazenará um valor dentro da escala do modelo de cor RGB (0 à 255 para cada uma das cores: *Red*, *Green* e *Blue* respectivamente). Um conjunto de *pixels* organizados, ou seja, *pixels* com informação de posição em espaço bidimensional constituem uma imagem.

Em um sistema RGB-D por sua vez, *pixels* estão organizados em um espaço tridimensional, ou seja, além de conter informação de cor (assim como um *pixel*), contêm ainda informações sobre coordenadas em um espaço tridimensional. Comumente, dá-se o nome de **ponto** ao *pixel* quando organizado em um espaço tridimensional, e **nuvem** de pontos ao conjunto de pontos.

Inicialmente, a maneira utilizada para aquisição dos dados foi utilizando a técnica de visão estereoscópica. Na Seção 3.1 é apresentado este assunto retirado do artigo Ridel, Neiva, Matsubara, Costa e Bogue (2012) com poucas adaptações.

3.1 Visão Estereoscópica

Diversos animais são capazes de estimar a profundidade de pontos no espaço a partir de duas imagens bidimensionais obtidas a partir de um par de olhos. Sistemas *stereo* são uma forma bastante conhecida de modelar o sistema de visão que estima a profundidade. Esse tipo de

sistema é composto por duas câmeras, em geral posicionadas entre si a uma distância bastante similar àquela existente entre os olhos do ser humano. A partir destas duas câmeras, duas imagens são obtidas: uma mais à esquerda e outra mais à direita. Dado um *pixel* que descreve alguma característica da cena na imagem da esquerda, é possível encontrar o *pixel* respectivo na imagem da direita. Ao realizar o alinhamento das duas imagens e ao mensurar a distância entre os pontos respectivos, é obtida a disparidade entre os dois pontos de vista. A distância entre estes dois pontos fornece a informação de profundidade. Objetos próximos ao observador apresentam maior disparidade, enquanto objetos distantes apresentam menor disparidade.

Apesar de ser aparentemente um problema trivial e de fácil compreensão, há diversas dificuldades envolvidas. Por exemplo, podem ocorrer alterações nas propriedades geométricas entre as imagens devido à distorções, diferença na visibilidade de um objeto entre duas imagens, presença de ruído, variações de condições de luz que podem se transformar em propriedades fotométricas (RZIZA, 2000), dentre outras. As implementações existentes de sistemas estereoscópicos de alta qualidade tentam minimizar esses problemas utilizando câmeras e condições de luz específicas para sistemas *stereo* de imagens. No entanto, nem sempre é possível ter um ambiente com luz controlada ou mesmo uma câmera de boa qualidade para o desenvolvimento de sistemas *stereo*s. O desafio está em utilizar câmeras de baixo custo, como *webcams*, e poder utilizá-las em quaisquer condições de luz. Para que isso seja possível é necessário realizar a maioria das correções via *software*.

Mapa de disparidade consiste em dada duas imagens provenientes de dois pontos de vista distintos, realizar a combinação entre elas e então estimar a profundidade. Para todo *pixel* encontrado em uma imagem, procura-se casá-lo com um *pixel* na segunda imagem. A priori é preciso ter conhecimento da distância e variação angular existente entre as duas imagens obtidas.

Para que a transformação de um sistema baseado em imagens 2D para uma imagem 3D tenha sucesso, é essencial que o mapa de disparidade retrate as distâncias o mais fiel possível, representando em escala de cinza uma imagem onde o branco reflete objetos mais próximos e o preto indica os mais distantes.

A implementação mais conhecida de sistema *stereo*, código livre, para imagens obtidas de câmeras comuns está na biblioteca do OPENCV (BRADSKI; KAEHLER, 2008). A biblioteca possui a função `FindStereoCorrespondenceBM` que gera um mapa de disparidade a partir de diferentes pontos de vista e uma série de parâmetros encapsulados na classe `CvStereoBMState`. Os implementadores da biblioteca procuram desenvolver funções de uso genérico e que possam tratar as mais diferentes situações, criando assim uma grande quantidade de parâmetros. Essa

facilidade acaba criando o problema de ajuste de parâmetros. O ajuste geralmente requer bons conhecimentos sobre cada um dos parâmetros, demandando tempo e conhecimento sobre a implementação.

A qualidade do mapa de disparidade, além dos fatores externos como luz e câmera, dependem também dos parâmetros. Algoritmos genéticos são populares por resolverem relativamente muito bem os problemas de ajuste de parâmetros. Nas próximas seções é apresentado o desenvolvimento de um conjunto de métodos que realizam a chamada da classe `CvStereoBMState` onde os parâmetros são ajustados automaticamente pelo algoritmo genético associado ao algoritmo DTW (*Dynamic Time Warping*) para um melhor desempenho.

3.1.1 Mapa de Disparidade

O conceito central da estereoscopia consiste na obtenção do mapa de disparidade. Diversas abordagens são encontradas na literatura para calculá-lo: Mühlmann, Maier, Hesser e Männer (2002) propõe o uso da técnica de divisão e conquista para casamento dos pontos; Scharstein e Szeliski (2003) propõe o uso de luzes estruturadas; Matthies, Kanade e Szeliski (1989) faz uso de uma técnica baseada no filtro de Kalman. Outras propostas surgiram com a modelagem do problema de cálculo de disparidades como um problema de otimização. A vantagem desta perspectiva é que permite a aplicação de técnicas já conhecidas a problemas de otimização: Han, Song, Chung, Cho e Ha (2001) e Gong e Yang (2002) propõe a utilização de algoritmos genéticos para a avaliação de mapas de disparidade.

A biblioteca OpenCV provê a `FindStereoCorrespondenceBM` para realizar o cálculo do mapa de disparidades. Esta função utiliza a abordagem de emparelhamento *block matching*. A ideia básica da técnica de emparelhamento consiste em dividir a imagem em pequenas regiões (blocos) (HUANG; ZHUANG, 1995; BEAUCHEMIN; BARRON, 1995), onde cada uma contém um número *x* de *pixels*. Uma vez que a imagem esteja dividida em blocos, tenta-se encontrar a correspondência dos *pixels* nos blocos de cada imagem. Embora rápida, podendo-se processar diversas imagens por segundo, se os parâmetros não forem ajustados corretamente os resultados apresentados são pobres. Para melhores resultados, ao total 10 parâmetros (9 inteiros e 1 ponto flutuante) devem ser configurados e fornecidos como entrada para a função `FindStereoCorrespondenceBM`. Entretanto, a escolha dos melhores valores para os parâmetros não é uma tarefa trivial. Suponha, por exemplo, um caso bem simples, onde todos os dez parâmetros são inteiros no intervalo de $0 - 30000$. Uma busca exaustiva em todos os possíveis estados significa testar 30000^{10} possíveis soluções, tornando-se computacionalmente inviável encontrar os parâmetros ótimos.

Ajuste de parâmetros pode ser visto como um problema de otimização, no qual temos uma função objetivo e restrições ao problema são apresentadas, ambas relacionadas às variáveis de decisão. Os valores destas variáveis de decisão são impostas pelas restrições a estas variáveis, formando um conjunto de soluções factíveis ao problema. Problemas de otimização são categorizados em problemas de minimização ou maximização, onde o ótimo global, será o menor (ou maior) valor possível para a função objetivo na qual o valor atribuído as variáveis de decisão não violem as restrições do problema.

Para a solução deste tipo de problema, algoritmos genéticos se apresentam como uma técnica que tem obtido bastante sucesso. Diversos algoritmos genéticos já foram desenvolvidos para problemas clássicos da área de otimização combinatória, como o problema do caixeiro viajante (LARRAñAGA, 1999) e da mochila (CHU; BEASLEY, 1998). A ideia consiste em utilizar algoritmo genético para ajustar os parâmetros da função do OPENCV para cálculos do mapa de disparidade e avaliar os resultados atingidos. Para avaliação mais ampla dos resultados, foi testado além da distância euclidiana, a utilização da medida de distância DTW.

Na busca por algoritmos capazes de melhorar a "acurácia" de mapas de disparidade, nesta seção são apresentados os métodos utilizados para o desenvolvimento deste trabalho.

Estereoscopia é um processo natural do ser humano. Com dois olhos é possível estimar a profundidade e proximidade de objetos. Isso só é possível devido a paralaxe de visualização entre as duas imagens captadas. A disparidade existente da pequena diferença de espaçamento entre os olhos permite que o cérebro humano processe ambas as imagens e seja capaz de estimar a profundidade dos objetos. Um sistema de câmeras *stereo* permite modelar o sistema de visão de um ser humano: duas câmeras separadas espacialmente por uma distância fixa, representando os olhos humanos, e um computador, simulando o processamento que nosso cérebro realiza sobre as imagens adquiridas.

Para garantir o correto emparelhamento de pontos, uma correta calibração do par *stereo* é fundamental para garantir o correto emparelhamento de pontos. Qualidade da câmera, distorções na lente e possíveis diferenças físicas são alguns fatores que influenciam a qualidade do processo. Mesmo que aparentemente as câmeras estejam visualmente alinhadas, é necessário retificá-las, ou seja, alinhá-las via *software*, para garantir a corretude e facilitar a correspondência dos pontos entre as duas imagens. Após alinhadas, é possível calcular a distância entre os *pixels* de ambas as imagens, e com isso obter o mapa de disparidade que indica a profundidade dos objetos da cena.

A função `FindStereoCorrespondenceBM` encontrada na OPENCV calcula o mapa de disparidade, utilizando a técnica de *block matching* e a soma da diferença absoluta, *sum of ab-*

solutte difference (SAD), para encontrar a correspondência entre as imagens. Três passos são necessários para o cálculo do mapa:

1. Uso de um filtro para normalizar o brilho da imagem e a textura.
2. Busca pela correspondência utilizando uma técnica de procura horizontal, visto que as imagens foram previamente alinhadas resultando em linhas de ambas imagens correspondentes.
3. Uso de um filtro para eliminar correspondências errôneas.

Além das duas imagens que a função recebe por parâmetro, é fornecido como parâmetro uma instância da classe `CvStereoBMState`, que contém treze parâmetros necessários para o cálculo do mapa de disparidade, são eles:

- *preFilterType*, *preFilterSize*, *preFilterCap* - atributos inteiros utilizados para normalizar o brilho e a textura das imagens, onde o *preFilterSize* determina o tamanho da janela à que as imagens serão normalizadas e possui tamanho que podem variar de 5×5, 7×7 (valor padrão), ... até 21×21. O parâmetro *preFilterCap* auxilia na terminação do pixel central na janela, onde o ponto central é computado por $min[max(I_c - \overline{I}, I_{preFilterCap}), I_{preFilterCap}]$, onde $\overline{I}$ é o valor médio da janela e *preFilterCap* é um número positivo limitado.
- *SADWindowSize*, *minDisparity*, *numberOfDisparities* - utilizados para o cálculo da correspondência calculada transladando o bloco de tamanho *SADWindowSize*, que pode variar da mesma forma que o atributo *preFilterSize*, além disso o parâmetro *minDisparity*,valor padrão 0, controla a correspondência, determinando onde a busca pelo casamento dos *pixels* deve iniciar. A busca pela disparidade é então realizada ao longo dos *numberOfDisparities pixels*.
- *textureThreshold*, *uniquenessRatio*, *speckleWindowSize*, *speckleRange* - para eliminar correspondências calculadas erroneamente, um filtro de limiar(*threshold*) é aplicado, onde se uma correspondência tem valor menor do que *textureThreshold*, com valor padrão 12, ela não é considerada. Visto que a correspondência utilizando *block matching* possui problemas com as bordas dos objetos, essa região gera áreas de grandes e pequenas disparidades chamadas *speckle*. Para diminuir essas regiões, é possível determinar um detector de *speckle* utilizando uma janela *speckle*, de tamanho *speckleWindowSize* e uma variação máxima aceitável para cada região, se as disparidades mínimas e máximas detectadas

estão dentro do valor *speckleRange*, a correspondência é aceita. O OPENCV faz uso da função de correspondência padrão usando o parâmetro de ponto flutuante, *uniquenessRatio*, valor padrão 12, e filtra os casamentos no qual a diferença do valor encontrado e do menor valor encontrado dividido pelo menor valor encontrado é menor do que o parâmetro *uniquenessRatio*.

- *preFilteredImg0*, *preFilteredImg1*, *slidingSumBuf* - três matrizes utilizadas pela função temporariamente para os cálculos internos.

A atribuição de valores para cada parâmetro pode fazer com que a imagem resultante varie drasticamente, é essencial atribuir cada parâmetro de forma que o mapa de disparidade gerado represente de maneira mais fiel possível, a variação de distância entre os objetos da cena.

3.1.2 Ajuste de Parâmetros

O objetivo se baseia em encontrar os melhores parâmetros para a classe `CvStereoBMState` com o intuito de gerar o melhor mapa de disparidade por meio da função `FindStereoCorrespondenceBM`. Para o problema apresentado foi desenvolvido um algoritmo genético para ajustar os parâmetros da classe `CvStereoBMState` da biblioteca OPENCV. Esses parâmetros são passados juntamente com as imagens esquerda e direita, previamente alinhadas, para a função `FindStereoCorrespondenceBM`, que constrói o mapa de disparidade. A primeira população do algoritmo genético é constituída de N elementos, cada elemento e_i, tal que $0 <= i < N$, possui um conjunto S, no qual S representa os parâmetros que devem ser ajustados, inicializados aleatoriamente. O conjunto S de e_i é utilizado como entrada para a função `FindStereoCorrespondenceBM`, e o mapa de disparidade gerado é comparado com a imagem *ground truth*, pela função avaliadora. O *ground truth* é uma imagem gerada computacionalmente por meio da técnica de luzes estruturadas. O *ground truth* representa o mapa de disparidade ideal para o par de imagens.

Para medir a similaridade entre o mapa de disparidade gerado e o *ground truth* e para o cálculo da diferença foram utilizadas duas abordagens, o algoritmo *Dynamic Time Warping* (DTW) e a distância euclidiana. Ambas calculam o $fitness$, número que representa a diferença entre a imagem obtida e a ideal. Quanto menor o $fitness$, melhor é o mapa de disparidade gerado pelo conjunto de parâmetros do elemento, por este motivo a população é ordenada crescentemente de acordo com o número gerado na função de avaliação, e os k melhores exemplos na população permanecem para a próxima iteração, enquanto os $N-k$ restantes, passam pelo processo de *crossover* e mutação.

Algoritmo 2: Algoritmo cruzamento

```
1 Seja I o conjunto dos todos os indivíduos.
2 Seja M o conjunto dos melhores indivíduos.
3 Seja S o conjunto dos 10 parâmetros de um indivíduo.
4 início
5     para cada indivíduo i ∈ I − M faça
6         para cada parâmetro p ∈ S do indivíduo i faça
7             Escolher aleatoriamente um indivíduo j ∈ M.
8             Atribuir ao parâmetro p do indivíduo i o valor do parâmetro p do indivíduo j.
```

A função de *crossover* do Algoritmo 2 tem como objetivo gerar novos indivíduos a partir de uma recombinação de parâmetros dos melhores elementos da população anterior.

Algoritmo 3: Algoritmo mutação

```
1 Seja I o conjunto dos todos os indivíduos.
2 Seja M o conjunto dos melhores indivíduos.
3 Seja S o conjunto dos 10 parâmetros de um indivíduo.
4 início
5     para cada indivíduo i ∈ I − M faça
6         Selecionar um conjunto C ⊆ S de k parâmetros aleatórios do indivíduo i.
7         para cada parâmetro p ∈ C do indivíduo i faça
8             Atribuir a p um valor gerado aleatoriamente.
```

O Algoritmo 3 insere valores aleatórios nos novos indivíduos gerados pelo cruzamento. Para cada indivíduo, são escolhidos aleatoriamente k parâmetros a serem alterados, e esses recebem um novo valor aleatório. O algoritmo termina quando o número máximo de iterações for atingido ou quando for encontrado um indivíduo na população cujo mapa gerado seja igual ao mapa ideal, ou seja, quando o $fitness$ for igual a zero. O resultado final é um conjunto de parâmetros para o `CvStereoBMState` tal que o mapa de disparidade gerado tenha o menor fitness entre todos dos conjuntos gerados pelo algoritmo genético.

3.1.3 Experimentos e Resultados

O algoritmo proposto recebe como parâmetro duas imagens, representando as imagens esquerda e direita, previamente alinhadas horizontalmente. Para cada par de imagens também foi fornecido um *ground truth*, ou seja, uma imagem representando o mapa de disparidade esperado a partir das duas imagens dadas. Para o funcionamento correto do algoritmo genético é necessário especificar o número N de indivíduos em uma população, o número it de iterações

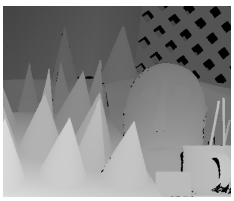
do algoritmo genético, a quantidade k de indivíduos considerados "melhores", e a quantidade de parâmetros p a serem alterados aleatoriamente na função de mutação. Para todos os experimentos foram utilizados os seguintes valores: número $N = 40$ de indivíduos em uma população, número escolhido com base em (GONG; YANG, 2002); número $it = 200$ de iterações do algoritmo genético; quantidade $k = 5$ de indivíduos considerados "melhores"; e quantidade de parâmetros $p = 3$ a serem alterados aleatoriamente na função de mutação. Estes valores foram escolhidos por meio de testes empíricos, onde foram avaliadas as soluções geradas. As imagens originais foram redimensionadas em 90% para que a quantidade de computação fosse reduzida. A variação dos números acima podem alterar o resultado final, assim como a inicialização aleatória da primeira população do algoritmo.

Para mostrar a efetividade do algoritmo, a técnica proposta foi aplicada em três imagens *stereo* bem conhecidas pela comunidade de visão computacional. O par de imagens *stereo* e o *ground truth* utilizados foram retirados de três bases de dados da Faculdade de Middlebury (HIRSCHMULLER; SCHARSTEIN, 2007) (SCHARSTEIN; SZELISKI, 2003) (SCHARSTEIN; PAL, 2007), a geração dos bancos de dados foi feita a partir da técnica de luzes estruturadas. Para a comparação entre as imagens obtidas e o *ground truth* por cada conjunto de parâmetros, duas abordagens foram consideradas: DTW e distância euclidiana, sendo possível comparar os resultados obtidos pelas duas funções avaliadoras.

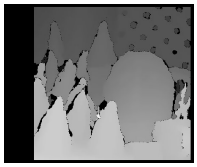
Na Figura 3.1 são apresentados os resultados obtidos com o conjunto de dados *Cones*. Inicialmente foi testada a chamada da função do mapa de disparidade com os parâmetros padrões 3.1 (e). Como pode ser observado, os contornos dos cones ficam com uma linha preta ao redor, o que significaria que essa região se encontra o mais distante da cena, o que não é verdade. Já na distância DTW (Figura 3.1 (d)) obtida pela abordagem proposta, já é possível perceber os cones onde os contornos ficam bem definidos. Além disso é possível perceber uma certa suavidade na transição das cores em escala de cinza, dando uma maior percepção de disparidade. Na distância euclidiana é possível perceber praticamente todos os cones da imagem chegando a apresentar os detalhes da grade ao fundo, a transição de cores também ocorre de maneira mais suave do que comparado ao resultado padrão.

A Tabela 3.1 representa a diferença calculada entre o mapa de disparidade gerado pelos parâmetros padrões e o *ground truth* utilizando as duas abordagem, DTW (DTW_P) e distância euclidiana (Euclidiana_P). Além disso são apresentados os *fitness* dos melhores mapas de disparidade gerados pelo algoritmo genético utilizando as funções de avaliação DTW(DTW_G) e distância euclidiana(Euclidiana_G). Quanto menor o valor melhor, entretanto só faz sentido comparar dois valores que utilizam a mesma métrica de distância.

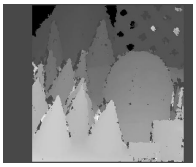
(a) Figura Original



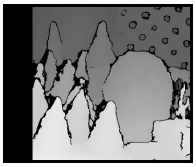
(b) GroundTruth



(c) DTW



(d) Euclidiana



(e) Padrão

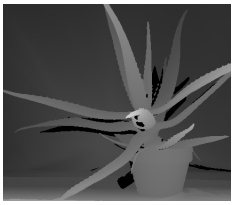
Figura 3.1: Conjunto de dados *Cones*

Tabela 3.1: Distância entre os mapas de disparidade gerados e o *ground truth* utilizando o *dataset Cone*.

Dataset	DTW_P	DTW_G	Euclidiana_P	Euclidiana_G
Cone	40786	18769	163515	81989

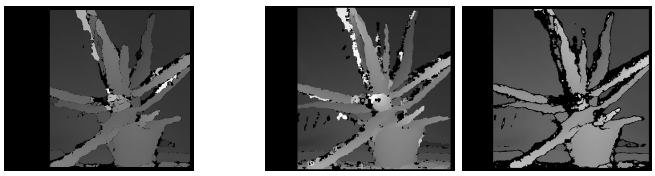
Para o conjunto de dados *Aloe* apresentado na Figura 3.2 ao utilizar os parâmetros padrões (Figura 3.2 (e)) o mapa de disparidade obtido, novamente deixa a desejar. Apesar de apresentar graduação de cores, essas não são feitas de forma harmoniosa, além de haver o problema das bordas, mostrando-se muito pretas. Utilizando a abordagem proposta (Figura 3.2 (c) (d)) já é possível distinguir as bordas e também a profundidade de cada uma das folhas. Neste conjunto a distância euclidiana consegue distinguir melhor alguns detalhes da planta como as folhas na vertical na parte central da figura. É possível perceber na abordagem euclidiana a disparidade é representada no plano de fundo, chão e parede, não sendo possível perceber essa diferença na imagem padrão. A Tabela 3.2 exibe os resultados para o conjunto *Aloe*.

(a) Figura Original



(b) GroundTruth

Finalmente na Figura 3.3 é apresentado o conjunto de dados *Moebius*. Entre os conjuntos testados, esta imagem é a que apresenta a maior quantidade de objetos. Assim como as imagens obtidas com os parâmetros padrão anteriores, o resultado nesse conjunto deixa desejar em



(c) DTW (d) Euclidiana (e) Padrão

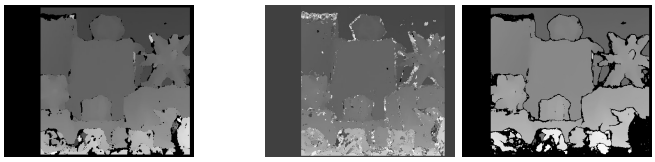
Figura 3.2: Conjunto de dados Aloe

Tabela 3.2: Distância entre os mapas de disparidade gerados e o *ground truth* utilizando o *dataset Aloe*.

Dataset	DTW_P	DTW_G	Euclidiana_P	Euclidiana_G
Aloe	502238	38439	133800	84438

relação aos limites de borda e suavidade nas cores. Os objetos menores são apresentados como manchas tornando-se muito díficil a distinção dos objetos. Na abordagem proposta (Figura 3.3 (c) e (d)) as formas dos objetos se apresentam de maneira mais fiel, além de representar melhor a distância, sendo possível perceber melhor por exemplo, o formato da estrela amarela. O resultado com o Distância Euclidiana (Figura 3.3 (d)) leva vantagem sobre as outras técnicas, apresentando um mapa de disparidade mais limpo se comparado ao resultado com o DTW, que peca ao apresentar diversas regiões pretas no objeto.

(a) Figura Original (b) Ground Truth



(c) DTW (d) Euclidiana (e) Padrão

Figura 3.3: Conjunto de dados *Moebius*

Na Tabela 3.3 são exibidos os resultados do conjunto *Moebius*. De acordo com as Tabelas 3.1, 3.2, 3.3, é possível verificar que os $fitness$ gerados pelo algoritmo genético (DTW_G e Euclidiana_G) se mostraram menores, ou seja, melhores do que os $fitness$ obtidos pela comparação entre o *ground truth* e o mapa de disparidade padrão, DTW_P e Euclidiana_P.

Tabela 3.3: Distância entre os mapas de disparidade gerados e o *ground truth* utilizando o *dataset Moebius*.

Dataset	DTW_P	DTW_G	Euclidiana_P	Euclidiana_G
Moebius	42111	24185	204673	64977

Como resultado desta Seção, foi desenvolvida uma abordagem para ajustar os parâmetros da função `FindStereoCorrespondenceBM` do OPENCV, utilizada para a criação do mapa de disparidade entre duas imagens. A `FindStereoCorrespondenceBM` é bastante sensível aos parâmetros e o ajuste correto para obtenção de um bom mapa de disparidade nem sempre é uma tarefa fácil. Para o ajuste dos parâmetros, foi proposto o uso de algoritmo genético que foi implementado utilizando duas medidas de distância: euclidiana e DTW. Os resultados experimentais mostram que a abordagem apresenta resultados satisfatórios para um problema de construção e mapeamento tridimensional e muito superiores aos parâmetros padrão da função. A abordagem foi testada com três conjuntos de dados bastante conhecidos na literatura, *Cones*, *Aloe* e *Moebius*. Os resultados mostraram que tanto a distância euclidiana quanto o DTW apresentam bons resultados. Na comparação entre DTW e distância euclidiana, o DTW apresenta resultados com menos ruídos e portanto foi considerado mais robusto que a euclidiana. Entretanto, mesmo com os resultados obtidos, a qualidade dos dados deixa a desejar. Dessa maneira, com o surgimento recente do *Kinect*, foi realizado um estudo sobre a aquisição dos dados utilizando câmeras do tipo *RGB-D*, mais notadamente *Kinect*. Na Seção 3.2 é apresentado um estudo sobre câmeras *RGB-D*.

3.2 *RGB-D*

RGB-D são sistemas de câmeras capazes de capturarem de forma sincronizada cor (*RGB - Red Green Blue*[1]) e profundidade *pixel* a *pixel* (D - *Depth*). Sistemas deste tipo evitam a complexidade de uma robusta computação de mapa de disparidade de sistemas *stereos*, e são muito mais rápidos que técnicas de escaneamento a *laser*.

[1]RGB é um modelo de cor independente de dispositivo: diferentes dispositivos detectam ou reproduzem um dado valor RGB diferentemente, uma vez que elementos de cores (tais como fósforo ou corantes) são responsáveis de forma individual por R, G e B sendo que seus níveis variam de fabricante para fabricante, ou até mesmo de dispositivo para dispositivo. Dessa forma, valores RGB não definem a mesma cor sobre dispositivos sem algum tipo de gerenciamento de cores.

3.2.1 Kinect

Kinect é um dispositivo lançado pela *Microsoft* em 2010. Este dispositivo incorpora um projetor de luz estruturada, uma câmera infravermelha (*IR*), uma câmera *RGB* além de outros periféricos como: uma sequência de microfones e um acelerômetro.

Embora seu objetivo inicial fosse a interação com jogos sendo apresentado inicialmente como um periférico para o console *Xbox 360*, posteriormente um conjunto de *drivers* oficiais foram lançados no ano de 2011 como parte do projeto *OpenNI* (DOTNETNUKE CORPORATION, 2012), consistindo de uma interface entre o computador e o *Kinect*, juntamente com uma biblioteca para se comunicar com o dispositivo e uma coleção de funções de visão computacional de alto nível. Além do *OpenNI*, existem ainda outros projetos conhecidos na área que permitem operar o *Kinect* como, por exemplo, o *Open Kinect* (BURRUS, 2012b) iniciado por Nicolas Burrus, a *Kinect SDK* lançada pela própria *Microsoft* (MICROSOFT CORPORATION, 2012a), entre outras.

As aplicações do *Kinect* são bastante amplas e atualmente, existem projetos que o utilizam tais como: uma interface de interação entre o homem e o computador para manipulação de imagens médicas (GALLO, 2011), um dispositivo para visão em robótica (HENRY, 2010), detecção de pessoas (XIA, 2011) e controle de vôo (STOWERS, 2011). Estes projetos buscam aproveitar da informação de "profundidade dos *pixels*" que o *Kinect* é capaz de fornecer, que é o seu diferencial de uma câmera comum.

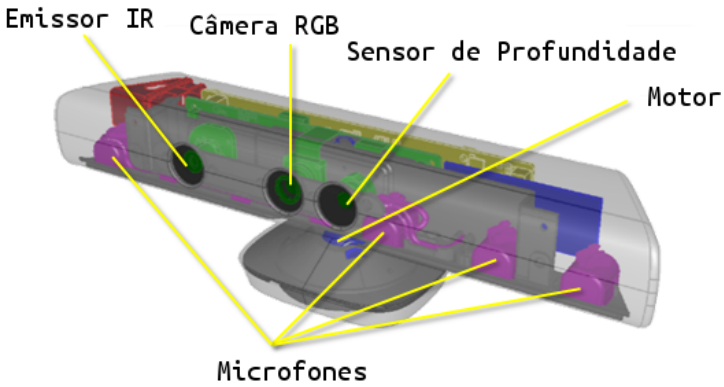


Figura 3.4: Visão geral do *Kinect* e de seus componentes. Basicamente o *Kinect* é composto de: uma câmera RGB, um conjunto de microfones, sensores capazes de detectar a profundidade e um motor (MICROSOFT CORPORATION, 2012b)

Na Figura 3.4 é apresentado uma visão geral do *Kinect* e seus componentes. Abaixo segue uma descrição de cada um dos recursos apresentados:

- Câmera RGB: imagens coloridas numa resolução de 640x480, com uma taxa de atualização de 30 quadros por segundo. Cada *pixel* é representado por *32 bits*, sendo que apenas 24 deles são usados (cada 8 armazenando o valor de um componente RGB);
- Emissor de luz infra-vermelha: emite uma padrão pseudo-aleatório de luz infra-vermelha;
- Sensor de Profundidade: um emissor de luz infra-vermelha espalha um padrão pseudo-aleatório sobre o ambiente e o sensor compara a deformação do padrão com um padrão gravado no *firmware*, estimando-se desta maneira o mapa de profundidades. A profundidade é armazenada como um valor de 16 *bits* e não representa uma distância de forma linear.
- Motor: permite mover o aparelho em alguns graus na vertical para alinhar a câmera;
- Microfones: um conjunto de 4 microfones são dispostos na parte inferior, permitindo um processamento de som mais avançado, por exemplo para melhorar sua qualidade, por meio de redução de ruído de ambiente e cancelamento de eco;

A câmera infravermelha, quando utilizada em conjunto com o emissor IR, é capaz de estimar a profundidade de um dado ponto na imagem relativa ao seu campo de visão.

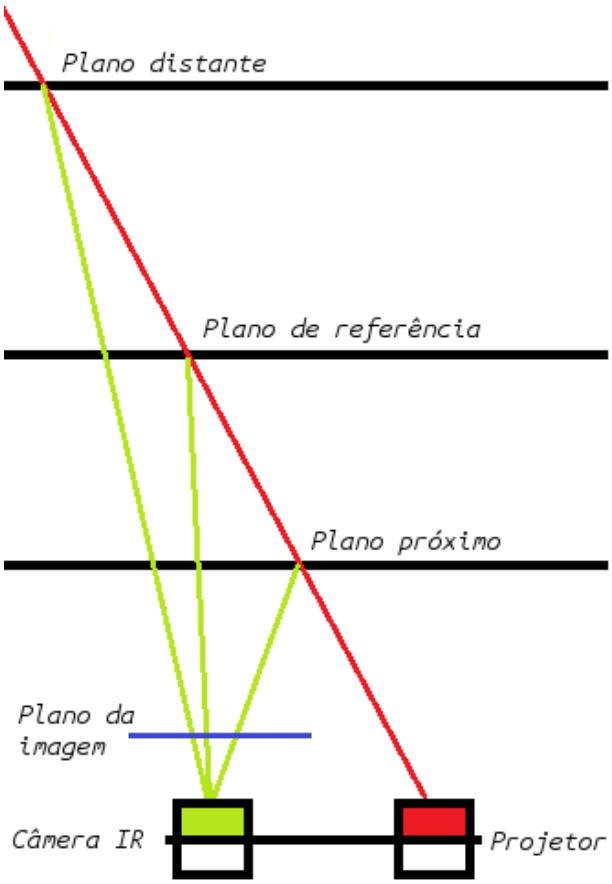


Figura 3.5: Modelo conceitual da estimativa de profundidade de forma de um único ponto de projeção (KJAER, 2011)

Na Figura 3.5 é apresentado um modelo conceitual da forma como o *Kinect* é capaz de estimar a profundidade de um ponto por meio de um conjunto de luzes estruturadas. Existem duas câmeras: uma à esquerda, que captura raios infravermelhos e uma à direita capaz de emitir raios infravermelhos.

No exemplo, o projetor está projetando apenas um único ponto no espaço (um padrão bastante simples), representado por uma linha vermelha. Este ponto é capturado pela câmera quando o raio atinge uma superfície. É possível observar a existência de três planos: um plano de referência, um plano mais próximo da câmera que o plano de referência e um plano mais distante da câmera que o plano de referência. Quando um ponto atinge a superfície no plano mais próximo, este ponto irá aparecer ligeiramente mais a direita da imagem do que se estivesse atingindo o plano de referência. Do mesmo modo, quando um ponto é projetado em um plano que está mais distante do plano de referência, o ponto irá aparecer mais a esquerda. Quando a origem e a direção da luz é conhecida de antemão, e a posição horizontal do ponto é conhecida por uma referência de profundidade, então é possível descobrir a profundidade da superfície que o ponto atinge baseado na posição horizontal no plano de imagem das câmeras.

Dado que as coordenadas *x* e *y* de qualquer ponto na imagem no plano é fornecida, obtendo-se também a profundidade nos permite compor a coordenada tridimensional do ponto. Assim, é possível extrair as coordenadas *3D* de cada um dos pontos da imagem. Realizando esta transformação em todos os pontos obtêm-se uma nuvem de pontos.

A partir da câmera *RGB* contida no *Kinect* é possível obter a cor de cada ponto (*24-bits RGB*). A resolução da câmera é de 640x480, portanto existem 640*480=307200 *pixels* por quadro. Isto implica um limite teórico de se obter no máximo 307200 pontos por quadro. Entretanto, na prática segundo os resultados obtidos por (KJAER, 2011), uma cena em boas condições de captura irá resultar em uma nuvem de pontos com no máximo 265000 pontos.

De acordo com Nicolas Burrus (um dos pioneiros em coletar informações do *Kinect* por meio de seus próprios experimentos), a profundidade de um ponto z pode ser calculada em metros por meio do "*raw disparity*" do ponto d (como provido pelo *Kinect*) utilizando a equação (BURRUS, 2012a):

$$z = 1.0/(d - 0.0030711016 + 3.3309495161) \tag{3.1}$$

onde d é um inteiro de *11 bits* podendo variar entre 0 à 2047. Este número representa uma quantidade teórica das possíveis disparidades de profundidades que o *Kinect* é capaz de representar.

Por meio de testes empíricos realizados por (KJAER, 2011), z irá mudar de sinal positivo para negativo quando d valer algo entorno de 1084. Portanto, valores além de 1084 não são utilizadas para medir a profundidade. (KJAER, 2011) constatou ainda que na prática o *Kinect* não é capaz de obter a profundidade de pontos que estejam a menos de $50cm$ de forma confiável. Isto significa que apenas quando d valer entre 434 e 1084 é que são valores mensuráveis de profundidade. Portanto, teoricamente existem apenas 650 pontos de disparidade. Destes possíveis valores, todos acima de 759 representam uma profundidade menor que 1 metro, implicando que metade de todas as possíveis disparidades de distâncias são utilizadas quando o *Kinect* se encontra a uma distância de $50cm$ a $1m$. Esta quantidade cai exponencialmente, de forma que apenas 16 disparidades podem ser representadas a uma profundidade entre 4 e 5 metros.

No próximo capítulo são apresentados técnicas de reconhecimento de padrões em representações tridimensionais.

4 *Reconhecimento de Padrões Tridimensionais*

Neste capítulo tem-se por objetivo apresentar um estudo sobre reconhecimento de padrões tridimensionais. Na Seção 4.1 é apresentada uma abordagem de reconhecimento de padrões utilizando o controle do *Wii*. A ideia central consiste na leitura dos dados dos sensores de movimentos (acelerômetros) como uma tripla (x, y, z) em um espaço tridimensional. O objetivo consiste na representação dos dados em três dimensões e posteriormente reconhecimento de padrões de movimentos.

Na Seção 4.2 é iniciado um estudo de reconhecimento de objetos em representações tridimensionais. É apresentado o conceito *pipeline* de aprendizado 3D, mostrando uma maneira de reconhecimento de objetos desde a obtenção dos dados até o reconhecimento propriamente dito. Dado que câmeras *RGB-D* são sistemas considerados novos, ainda existem poucos métodos propostos especificamente para aquisição de objetos utilizando este tipo de equipamento. Neste trabalho é apresentado um modelo *pipeline* para aquisição de objetos baseado em modelos de aquisição de objetos a partir de câmeras *stereo*. Nas próximas seções abordados os três primeiros passos do *pipeline*: *Downsampling* na Seção 4.2.2 , Segmentação na Seção 4.2.3 e por último, Anotação na Seção 4.2.4.

Na próxima Seção é apresentada uma abordagem de reconhecimento de movimentos utilizando o *Wii*, baseado no artigo Bogue, Matsubara e Bessa (2012).

4.1 Uma Abordagem de Reconhecimento de Movimento utilizando o Wii

A utilização de sensores de movimentos como acelerômetros e giroscópios vem alterando significativamente a interação homem-máquina em dispositivos, como câmeras fotográficas, *notebooks*, celulares, *tablets* e video-games. Por exemplo, movimentos bruscos podem causar avarias em discos rígidos e dispositivos de *cd-rom* em *notebooks*, de forma que sensores de mo-

vimento são utilizados para minimizar o risco de que isto ocorra. Celulares e *tablets* possuem sensores de movimento para criar interfaces mais intuitivas para a apresentação de imagens e jogos. Video-games fazem uso de controles com diversos sensores de movimentos para criar jogos nos quais a interface exige do jogador movimentos mais próximos do movimento real para uma melhor interatividade, imersão no jogo e entretenimento do jogador. Apesar do reconhecimento de movimentos ser realizado nesses exemplos com relativo sucesso, a detecção de movimentos e gestos ainda é um desafio interessante. Há ainda a necessidade de se desenvolver novos algoritmos, que sejam mais rápidos e precisos.

O problema de reconhecimento de movimentos está fortemente relacionado ao problema de reconhecimento de padrões, um tópico amplamente estudado em Aprendizado de Máquina. Reconhecimento de pessoas, reconhecimento de caracteres, mineração de dados, análise de crédito, problemas de classificação de *spam* e de genes são alguns exemplos nos quais o reconhecimento de padrões é empregado fazendo uso dos algoritmos de aprendizado de máquina. A maioria desses problemas possui uma característica em comum: entrada dos dados no formato de tabelas atributo-valor. Contudo, no problema do reconhecimento de movimentos a entrada normalmente é uma série temporal. Assim, é preciso adaptar os algoritmos de Aprendizado de Máquina existentes para utilizá-los no reconhecimento de movimentos.

Neste sentido, torna-se necessário considerar os algoritmos para o tratamento de séries temporais, dentre os quais um dos mais conhecidos é o DTW (*Dynamic Time Warping*) (KEOGH; RATANAMAHATANA, 2005a). Tal algoritmo compara duas séries temporais, construindo uma matriz que representa um mapa de alinhamento entre as duas. O último valor inserido na matriz representa a quantidade de distorções (ou *warpings*) necessárias para a conversão de uma série na outra. Este valor satisfaz os axiomas necessários para métricas de distância, notadamente: não negatividade (d(x, y) $>=$ 0), identidade (d(x, y) = 0 $\Leftrightarrow$ x = y), simetria (d(x, y) = d(y, x)) e desigualdade triangular (d(x, z) $<=$ d(x, y) + d(y, z)). Assim, o DTW fornece uma medida de distância entre duas séries temporais, adequada para ser utilizada em algoritmos baseados em distâncias como o KNN. Desta forma, é possível obter uma variação de KNN que trata séries temporais sem maiores modificações. No entanto, o algoritmo obtido desta forma herda as qualidades e defeitos do KNN. E um dos problemas de KNN está no tempo de classificação de novos exemplos quando o conjunto de treinamento é grande. Para lidar com tal problema, é possível usar o TRKNN (*Template Reduction k-Nearest Neighbors*), que reduz o conjunto de treinamento armazenado para alguns poucos exemplos mais signicativos, reduzindo assim o tempo de classificação de novos exemplos (movimentos) em até três vezes, sem perda de qualidade em termos de taxa de acerto.

4.1.1 Abordagens de Reconhecimento de Movimento

Em (BILLON, 2008), os autores propõem um método de reconhecimento de gestos em tempo real. A ideia central do método consiste na transformação do movimento em uma espécie de assinatura artificial. Esta assinatura é então decomposta em três partes: começo, meio e fim. Com o auxílio da técnica DTW, em tempo real, o método detecta se o movimento sendo realizado é compatível com o início de algum movimento existente e desse modo procura casá-lo com o *template* de um movimento treinado. Além disso, para o desenvolvimento deste método, (BILLON, 2008) utilizam propriedades do PCA (*Principal Components Analysis*). PCA é uma técnica estatística muito útil para gerar um espaço *2D* dedicado para cada gesto, encontrando-se aplicações em campos como reconhecimento de face e compressão de imagem, sendo também uma técnica comum para encontrar padrões em dados de alta dimensão. No reconhecimento de movimentos, um mesmo gesto realizado pode diferir no espaço temporal. Isso significa, por exemplo, que embora dois movimentos sejam semelhantes e representados por uma curva similar, estas curvas podem diferir grandemente quando dispostas no espaço temporal. Para comparar curvas e outras formas que estejam fora de sincronia temporal, é utilizado a técnica DTW, que corresponde a uma técnica de casamento de padrão utilizada para comparar sinais, não necessariamente do mesmo tamanho, baseado nas característica da forma (POLI, 2007). A ideia básica consiste em comparar amostras de sinais de entradas desconhecidos com exemplos de um conjunto de sinais *template*. A técnica DTW é um algoritmo capaz de medir a semelhança entre duas sequências que variam em tempo e velocidade. Esta técnica mostra-se bastante robusta quando aplicada a séries temporais, permitindo que formas similares sejam casadas mesmo se estiverem fora de sincroniza temporal (KEOGH; RATANAMAHATANA, 2005a). Devido a esta flexibilidade, DTW é largamente utilizado na ciência, medicina, indústria e finanças.

Outra abordagem para o reconhecimento de movimentos (SCHLOMER, 2008) utiliza uma técnica que se baseia na utilização do processo Modelos Ocultos de Markov (*Hidden Markov Model* - HMM), que segundo Schlomer, Poppinga, Henze e Boll (2008) é um método já conhecido em reconhecimento de movimentos e oferece resultados confiáveis para padrões com variação espacial e temporal. São ainda utilizados: redes neurais recorrentes, permitindo o reconhecimento de movimentos contínuos (MURAKAMI; TAGUCHI, 1991) e inclusive a utilização de uma Máquina de Estados Finita (*Finite State Machine* - FSM) capaz de reconhecer movimentos (HONG, 2000). A proposta da utilização da Máquina de Estados Finita por (HONG, 2000) para reconhecimento de movimento consiste na utilização de um *clustering* espacial e alinhamento temporal, onde cada gesto é definido como uma sequência ordenada de estados

no espaço temporal-espacial. A informação temporal é então futuramente integrada para construir uma Máquina de Estados Finita capaz de reconhecer movimentos. Existe ainda a aplicação de métodos como: filtro de partículas e condensação, modelos conexionistas (MITRA; ACHARYA, 2007) e por último KNN (KIM, 2008) (ZHANG, 2006).

KNN (*k-Nearest Neighbors*) é um algoritmo bastante comum utilizado para classificação. A classificação é baseada na proximidade com que a nova entrada está dos exemplos de treinamento no espaço. É um classificador simples e rápido (KIM, 2008). Uma das deficiências do KNN está relacionada a quantidade de exemplos de treinamento. Sendo um algoritmo "*lazy*" que simplesmente memoriza o conjunto de exemplos de treinamento, quanto maior o conjunto de treinamento armazenado, mais lento é a classificação de novos exemplos. Uma proposta promissora foi apresentada em (FAYED; ATIYA, 2009) na qual os exemplos que possuem pouca ou nenhuma influência na classificação de novos exemplos são eliminados, tornando assim a técnica mais adequada para problemas que necessitam de uma rápida resposta na classificação.

4.1.2 Modelo de Reconhecimento

Com base nos acelerômetros do controle do video-game nintendo *Wii*, conhecido também como *wiimote*, foram armazenados para cada movimento dezenas de pontos, sendo que cada ponto é composto pela aceleração dos eixos x, y e z. A combinação destes pontos gera uma curva no plano *3D*, que caracteriza um determinado movimento. Para que fosse obtido uma boa precisão, foi necessário armazenar o mesmo movimento repetidas vezes, pois é difícil para o usuário repetir o gesto com exatidão.

O algoritmo de DTW é utilizado em conjunto com o método TRKNN, de forma que o primeiro forneça a verosimilhança (distância) entre dois movimentos ao TRKNN. Figura 2.4 exibe-se a construção de duas *Nearest Neighbor Chain* utilizando a técnica DTW como medida de verosimilhança. Neste exemplo, iniciando-se pela sequência x_1 da classe positiva, o algoritmo encontra que o vizinho mais próximo de x_1, que seja da classe negativa, é x_{11}. A sequência x_{11} verifica que a sequência positiva mais próxima é x_{12} e x_{12} então localiza que a sequência negativa mais próxima é x_{11}, finalizando a construção da *Nearest Neighbor Chain* de x_1. A sequência x_{ij} na cadeia será eliminada se $d_{ij} > \alpha * d_{i,j+1}$, onde α é um limiar > 1 e $j = 0,2,4...$ até o tamanho da cadeia.

O Algoritmo 4 exibe um pseudocódigo da abordagem desenvolvida utilizando DTW + TRKNN.

Algoritmo 4: Algoritmo Reconhecimento de Movimentos

```
1  início
2      para cada sequência x_i do conjunto de treinamento faça
3          Construir a Nearest Neighbor Chain de x_i
4          Remover sequências x_ij se d_ij > a * d_i,j+1
5      para cada sequência t do conjunto de teste faça
6          Selecionar as k sequências mais próxima a t.
7          para toda classe c das k sequências mais próximas faça
8              Agrupar as n sequências de k pertencentes a classe c.
9              Calcular w_c = Σ_{i=1}^{n} 1/d^2
10         Rotular a sequência t com a classe de maior peso w_c.
```

4.1.3 Avaliação Experimental

A avaliação experimental tem como objetivo mostrar duas importantes características do método: alta taxa de acerto e rápida resposta de classificação.

Como mencionado, para avaliação do método proposto foi utilizado o *wiimote*. Uma das facilidades ao utilizar o *wiimote* está no uso de `bluetooth` como protocolo de comunicação. O uso de *bluetooth* possibilita a conexão do *wiimote* com computadores pessoais sem maiores dificuldades. A fácil comunicação com computadores pessoais motivou o desenvolvimento de bibliotecas de desenvolvimento que permitem um fácil acesso a leitura dos botões e também dos acelerômetros, sensores de movimentos, nos eixos três eixos: x, y e z. No desenvolvimento deste trabalho foi utilizado a biblioteca *wiiusej*, disponível na internet pelo site http://code.google.com/p/wiiusej/.

Para a avaliação experimental do método proposto foi constituído um conjunto de dados obtido pela avaliação de 12 movimentos[1] (baixo, cima, círculo, diagonal inferior esquerda, diagonal inferior direita, diagonal superior direita, diagonal superior esquerda, direita, esquerda, quadrado, tenis e z). Cada movimento foi realizado 100 vezes por três diferentes voluntários de maneira alternada, totalizando 1200 movimentos que em média são compostos por um conjunto de 56 pontos cada.

Sabe-se que o algoritmo KNN tem um desempenho interessante para o reconhecimento de problemas temporais (KEOGH; RATANAMAHATANA, 2005a), entretanto este algoritmo não tem resposta rápida quando o conjunto de treinamento é grande. Para solucionar esta limitação do KNN propõe-se a união de DTW e TRKNN, este último proposto em (FAYED; ATIYA, 2009) que reduz o tempo de classificação de novos exemplos sem uma perda significativa de

[1] base dísponível em www.lia.facom.ufms.br

desempenho em termos de taxa de acerto. Deste modo esta seção procura responder as seguintes questões:

Avaliação 1: Qual é a taxa de acerto do método TR-KNN e KNN "puro"?

Para este experimento foram avaliados os 12 tipos de movimentos com 100 exemplos cada um. Com um conjunto de 1200 exemplos, os algoritmos foram avaliados com validação cruzada de 10 partições e foram utilizados diferentes valores de *k* para avaliação das abordagens *TRKNN* (com configuração de limiar com valor 1.2 para a fase de condensação) e KNN. Os resultados são apresentados na Tabela 4.1, onde cada linha representa os movimentos testados e as colunas representam a taxa de acerto do KNN sem redução da quantidade de instâncias armazenadas e do KNN com a redução pela técnica *TRKNN*, os valores entre parênteses representam o desvio padrão.

Tabela 4.1: Taxa de acerto obtida utilizando validação cruzada de 10 partições para cada tipo de movimento utilizando TR-KNN e KNN

Movimento	**1-NN**	**TR-1NN**	**3-NN**	**TR-3NN**	**5-NN**	**TR-5NN**
Baixo	100% (0,0)	95% (10,80)	97% (9,48)	97% (9,48)	97% (9,48)	97% (9,48)
Cima	100% (0,0)	100% (0,0)	100% (0,0)	99% (3,16)	100% (0,0)	99% (3,16)
Círculo	97% (9,48)	99% (3,16)	97% (6,74)	98% (6,32)	96% (8,43)	95% (8,49)
Diag. Inf. Esq.	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)
Diag. Inf. Dir.	92% (17,51)	89% (17,91)	91% (20,24)	83% (29,07)	90% (21,60)	78% (30,84)
Diag. Sup. Dir.	97% (9,48)	96% (12.64)	97% (9,48)	96% (12,64)	97% (9,48)	94% (15,77)
Diag. Sup. Esq.	100% (0,0)	99% (3,16)	100% (0,0)	99% (3,16)	100% (0,0)	95% (15,81)
Direita	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)
Esquerda	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)
Quadrado	96% (12,64)	96% (0,0)	96% (12,64)	96% (12,64)	100% (0,0)	96% (12,64)
Tennis	97% (4,83)	91% (19,11)	94% (9,66)	87% (27,90)	92% (15,49)	86% (27,96)
Z	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)	100% (0,0)

A taxa de acerto para os 12 movimentos são próximos de 100% em quase todos os resultados. Nos 72 resultados obtidos utilizando a combinação de DTW e KNN, 61 possuem uma média de taxa de acerto maior ou igual a 95%. Apenas os resultados do movimento de Diagonal Inferior Direita e Tennis possuem resultados abaixo de 95%. Não existe uma explicação exata para o resultado abaixo da média destes dois movimentos. Os resultados obtidos mostram que o uso conjunto de DTW e KNN são bastante promissores com altas taxas de acerto para os movimentos testados. O TRKNN reduz a quantidade de exemplos comparados no momento de classificação. Uma característica desejada é que o algoritmo reduza a quantidade de exemplos sem a redução da taxa de acerto. Assim essas observações induzem as próximas duas perguntas da avaliação experimental realizada por este trabalho.

Avaliação 2: O uso de TRKNN reduz significativamente a quantidade de exemplos?

Conforme exposto na Tabela 4.2, após o processamento do método TRKNN, a redução obtida foi de 1200 para 338,6 (desvio padrão = 18,36) na média para as 10 partições. A redução foi de aproximadamente 70% do número de exemplos de avaliação do algoritmo KNN.

Tabela 4.2: Média e desvio padrão do número de exemplos de treinamento armazenados do KNN e do TRKNN

KNN	TRKNN
1200 (0,0)	338,6 (18,36)

A redução de cerca de 70% dos exemplos pode impactar no algoritmo de maneira negativa na sua taxa de acerto. Para realizar uma verificação sob a perspectiva de um teste estatístico de significância a avaliação seguinte realiza o seguinte questionamento.

Avaliação 3: A redução do número de exemplos armazenados implica em redução da taxa de acerto de maneira significativa sobre a perspectiva de um teste estatístico? Para responder esta pergunta de maneira devida foram realizados testes-t pareados comparando KNN e sua respectiva versão utilizando TRKNN. A Tabela 4.3 apresenta os valores médios dos algoritmos juntamente com os desvios padrões entre parênteses.

Tabela 4.3: Média e desvio padrão da taxa de acerto obtidos utilizando validação cruzada de 10 partições utilizando variações de KNN e TRKNN

1NN	TR-1NN	3NN	TR-3NN	5NN	TR-5NN
98.16% (2,47)	97,08% (2,49)	97.66% (2,96)	96,24% (3,66)	97,33% (3,32)	94,99% (4,04)

Embora em alguns casos houve redução na taxa de acerto e/ou aumento do desvio padrão quando a técnica TRKNN foi aplicada, utilizando um intervalo de confiança de 95% e 11 graus de liberdade não foi possível detectar diferença significativa em nenhuma das comparações entre KNN e a respectiva versão TRKNN nos experimentos realizados. Desse modo, apesar do método reduzir a quantidade de exemplos em aproximadamente 70% na média não há perda de desempenho significante em termos de taxa de acerto.

Sem perda de desempenho é interessante verificar a redução do tempo de execução do método. Esta verificação é feita na avaliação a seguir.

Avaliação 4: Existe redução significativa no tempo de execução da classificação utilizando TRKNN em comparação com o KNN?

Na Tabela 4.4 são apresentados os tempos de execução dos experimentos em segundos (em uma máquina com processador *Core 2 Duo*) comparando o tempo de classifi-

cação (excluindo o tempo de treinamento) para todos os experimentos apresentados na Tabela 4.1 para KNN e TRKNN, ou seja, somando os tempos de classificação para cada método com *K* igual a 1, 3 e 5. Assim os 26,45 segundos seria o tempo de classificação de 3600 movimentos o que resultaria em 0,00734 segundos para classificar cada movimento utilizando TRKNN. Assim pode-se concluir que para a conjunto de dados apresentado o TRKNN é pelo menos três vezes mais rápido que o KNN tradicional.

Tabela 4.4: Média e desvio padrão do tempo de classificação utilizando KNN e TRKNN

KNN	**TRKNN**
90,33s (3,15)	26,45s (2,51)

Neste trabalho, além do *wiimote* para aquisição de dados, é utilizado também o *Kinect*. Embora a aquisição de dados em ambos os equipamentos resultem em uma representação tridimensional, a semântica é um pouco diferente. Enquanto que no `wiimote` fora realizado a leitura dos acelerômetros como uma tripla (x, y, z) o *Kinect* fora utilizado de forma a capturar uma imagem em três dimensiões, ou seja, ao invés de capturar uma imagem em duas dimensões, com o *Kinect* é possível capturar também profundidade, resultando portanto em uma imagem em três dimensões. Neste contexto de semântica, na próxima seção é apresentado um estudo sobre reconhecimento de objetos tridimensionais, com foco na utilização dos dados adquiridos pelo *Kinect*.

4.2 Reconhecimento de Objetos Tridimensionais

O fundamento básico em reconhecimento de objeto está na capacidade de se verificar a correspondência entre dois objetos. Dada uma cena consistindo de um ou mais objetos, é possível identificar e localizar estes objetos que são suficientemente visíveis pelos sistemas sensoriais?

Reconhecimento de objeto confiável, estimativa de pose e rastreamento (*tracking*) são tarefas consideradas críticas em robótica (TAYLOR; KLEEMAN, 2003) (EKVALL, 2005) (ZICKLER; VELOSO, 2006) (MITTRAPIYANURUK, 2004). O componente chave de muitos sistemas de visão inteligentes é principalmente a capacidade de reconhecer um objeto.

Há cerca de 40 anos atrás foram iniciadas pesquisas em visão computacional buscando resolver o problema reconhecimento de um objeto *3D* utilizando uma única imagem *2D*. Uma vez que humanos realizam esta tarefa com pouco esforço, acreditava-se que esta tarefa seria realizada facilmente. Porém, na prática este problema não possui uma solução definitiva e continua em aberto.

O problema de reconhecimento de objetos pode ser subdividido em categorias baseados na dimensionalidade da sua descrição espacial:

1. reconhecimento de objetos *2D* a partir de imagens *2D* (COLLET, 2009);
2. reconhecimento de objetos *3D* a partir de uma única imagem *2D*;
3. reconhecimento de um objetos *3D* a partir de um modelo *3D* (MOKHTARIAN, 2001) (CHEN; BHANU, 2009) (LAI, 2011b);
4. reconhecimento de objetos *2D* e *3D* a partir de múltiplas imagens *2D* obtidas de diferentes pontos de vista (KIRBY; SIROVICH, 1990) (TURK; PENTLAND, 1991);

A proposta deste trabalho se encaixa no item 3. Dentro de um ambiente *3D*, será realizada a correspondência entre dois objetos: o objeto que foi treinado (a partir de um modelo *3D*) e aquele objeto que está sendo observado no momento (a partir de uma imagem *3D*). É importante que estas características sejam invariantes à rotação e translação, e preferencialmente invariante a mudanças na iluminação e ponto de vista da câmera *3D*.

Em reconhecimento de objetos *3D*, a tarefa consiste em atribuir uma classe para cada objeto solicitado. As possíveis classes que podem ser atribuídas a um objeto são conhecidas previamente. Segundo Lai, Bo, Ren e Fox (2011a) o estado da arte de abordagens para realizar esta tarefa são geralmente sistemas de aprendizado supervisionado. Um conjunto de objetos são anotados com suas respectivas classe e então são fornecidas a um classificador, que aprende um modelo que permite distinguir entre as diferentes classes.

A correspondência entre dois objetos geralmente é feita a partir de características extraídas (LOWE, 1999) (SCHMID; MOHR, 1997) (BARIYA; NISHINO, 2010) (ZHONG, 2009) (FROME, 2004). Um grande número de características podem ser obtidas com algoritmos eficientes (algumas dessas características foram apresentadas na Seção 4.2.5). Para minimizar o custo da extração destas características, (LOWE, 2004) propõe uma abordagem de filtro em cascata, no qual as operações mais caras são aplicadas apenas em objetos que já passaram por testes iniciais, ou seja, são objetos candidatos.

Para a avaliação de técnicas de reconhecimento de objetos é necessário abordá-las sob uma perspectiva em dois níveis (LAI, 2011a):

- *Category level*: Reconhecimento e detecção que envolve classificar objetos nunca vistos anteriormente como pertencentes a mesma categoria de objetos vistos anteriormente;

- *Instance level*: Reconhecimento e detecção identifica se um objeto é fisicamente o mesmo objeto que foi previamente visto;

A Figura 4.1 apresenta uma visão geral do *pipeline* de aprendizado. Abaixo são listados e enumerados os passos do *pipeline*:

1. Os dados são obtidos;
2. É aplicada a técnica de *downsampling* para que seja reduzida a quantidade de pontos armazenados (Seção 4.2.2);
3. A imagem é segmentada (Seção 4.2.3);
4. É realizada a anotação em cada um dos segmentos gerados na fase anterior. Além disso, um objeto pode ser composto de vários outros subobjetos (Seção 4.2.1);
5. Características que permitam identificar um objeto são extraídas (Seção 4.2.5);
6. As informações referentes a um objeto são armazenadas como um objeto em *Python*;

4.2.1 Modelo de Representação Proposto

A proposta deste trabalho está em uma abordagem de representação de conhecimento de objetos 3D. O objetivo não consiste em uma nova organização baixo nível, mas sim uma organização em alto nível. A abordagem de representação proposta utiliza os conceitos de orientação a objetos, sendo composta de duas propriedades: características e comportamento, representado respectivamente por seus atributos e métodos.

Na Seção 4.2.1 é apresentado o modelo conceitual de representação, juntamente com os aspectos relativos à implementação do modelo.

Modelo Conceitual

O conceito de orientação a objeto surgiu no MIT (*Massachusetts Institute of Technology*) no final dos anos 1950 e início dos anos 1960 (MCCARTHY, 1960) (MCCARTHY, 1962). Neste período, dentro do ambiente de inteligência artificial, objetos eram referidos como itens identificados (LISP *atoms*) com propriedades (atributos). Este conceito ainda se mantêm presente em orientação a objetos.

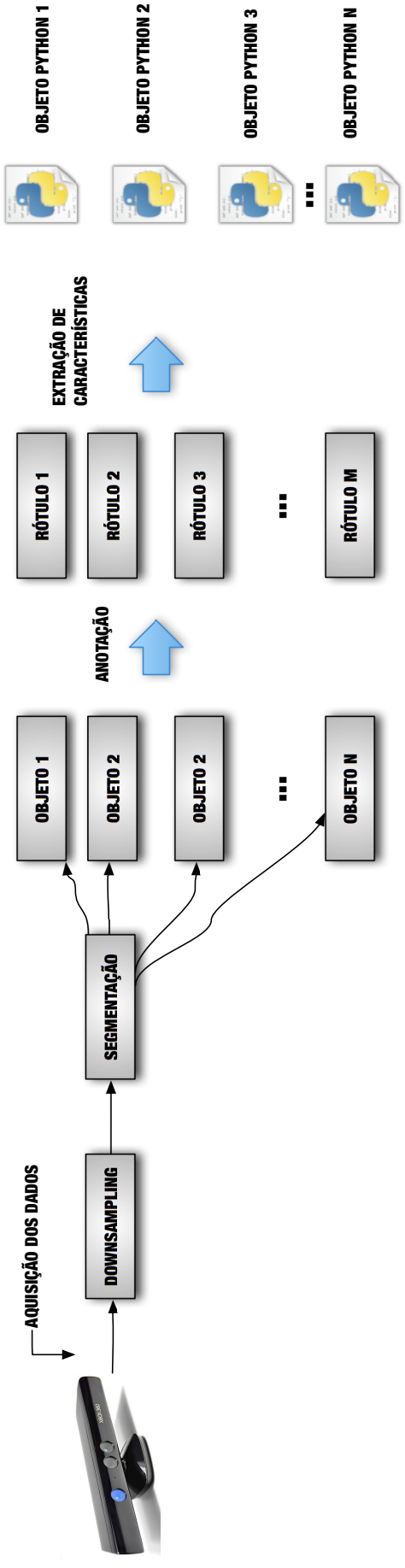


Figura 4.1: *Pipeline* de aprendizado *3D* proposto.

O modelo proposto consiste em representar cada objeto concreto ou abstrato como um objeto computacional. Assim como acontece no mundo real, objetos podem estar isolados ou relacionados com outros objetos.

Nas próximas seções são apresentados de forma mais abrangente os dois conceitos fundamentais do modelo: objeto e relacionamento entre objetos. Na Seção 4.2.1 são apresentados os aspectos de implementação do modelo proposto: a linguagem de programação utilizada, aspectos e decisões de implementação, entre outros assuntos.

Objeto

Segundo os conceitos de Orientação a Objetos, um objeto é um modelo computacional de uma entidade concreta ou abstrata, representado por suas características e comportamento implementados como atributos e métodos, respectivamente.

Um objeto encapsula os dados, ou seja, sua verdadeira representação do modelo computacional está escondida do "ambiente externo". Na abordagem deste trabalho, um objeto será representado internamente por: uma nuvem de pontos (x, y, z), uma nuvem de pontos de vetores normais, e possíveis características que identifiquem unicamente este objeto (como: VFH - *Viewpoint Feature Histogram*, explicado com detalhes na Seção 4.2.5, uma *bounding box* que estime aproximadamente o seu tamanho, um valor em uma escala métrica de cores que represente uma aproximação da cor do objeto, etc.).

Suponha, por exemplo, que se deseje representar um *notebook*, conforme apresentado na Figura 4.2. Um *notebook* é um objeto. Para representar o objeto (neste caso concreto), será criado um objeto computacional. Este objeto computacional irá encapsular todos os dados que representam o *notebook* (nuvem de pontos, nuvem de pontos das normais, VFH, etc). Não é de interesse que entidades externas ao objeto tenham conhecimento de sua representação, mas sim na interação que este provê, ou seja, qual a sua interface pública.

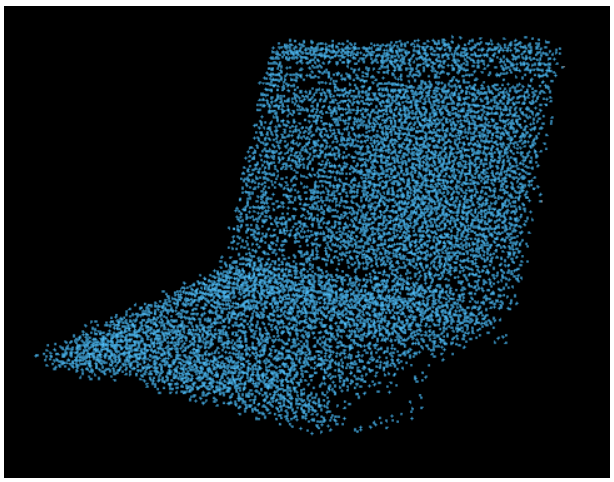


Figura 4.2: Exemplo de uma nuvem de pontos representando um *notebook*

A princípio, o objeto é capaz de realizar duas tarefas principais: compará-lo com outro objeto e estimar a distância entre si mesmo e outro objeto no espaço. Estes dois comportamentos são implementados como métodos e, em um primeiro momento, permitem que o sistema aprenda o conceito de um objeto e seja capaz de localizá-lo em um ambiente. Além disso, é possível estimar a distância em que este se encontra da câmera ou a distância entre dois objetos quaisquer que estejam no mesmo espaço. Dado o *notebook*, por exemplo, ela deve provêr métodos que permitam compará-la com outro objeto qualquer e também mensurar a distância entre ela e outro objeto.

Relacionamento entre Objetos

O modelo irá permitir a representação dos dois relacionamentos mais comuns entre objetos: ***"IS A"*** ('é um') e ***"HAS A"*** ('tem um'). Em orientação a objeto, o conceito de relacionamento ***"HAS A"*** é representado pela composição ou agregação de objetos. Uma vez que agregação é um conceito difícil de ser distinguido de associação, neste trabalho será utilizado apenas o conceito de composição entre objetos (composição é uma relação mais forte que agregação).

Composição de objetos é um modo de compor objetos mais complexos a partir de objetos mais simples. Composição é o bloco básico de construção de diversas estruturas de dados, incluindo lista ligada e árvore binária por exemplo.

A distinção entre os conceitos de composição e especialização é que este último consiste no processo de adicionar detalhes a partir de um objeto mais geral, criando-se assim um tipo mais específico. Especialização é representado pelo conceito ***"IS A"***, correspondendo ao relacionamento de especialização (ou herança). Um exemplo do mundo real de composição pode ser

visto a partir da observação que um automóvel é composto de objetos menores, por exemplo: um automóvel "tem um" chassi. Em um relacionamento de composição, temos o objeto pai (neste caso o automóvel) e o(s) objeto(s) filho(s) (neste caso o chassi).

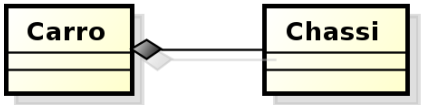


Figura 4.3: Representação de uma relação de composição. Um Carro "tem um" Chassi

No relacionamento de herança, existe o conceito de objeto mais genérico e objeto mais específico. Por exemplo, um cachorro é um mamífero (neste caso, o objeto mais genérico seria o mamífero e o objeto mais específico seria o cachorro), um gato é um mamífero, um carro é um veículo, um caminhão é um veículo, etc. No caso do gato e do mamífero, o gato é uma especialização do mamífero. Mamífero contêm as características mais genéricos da espécie, enquanto que o gato, além de herdar todos as características de mamífero, contêm conceitos adicionais que o especializam. Na Figura 4.4 é apresentado a representação da relação ***"IS A"***.

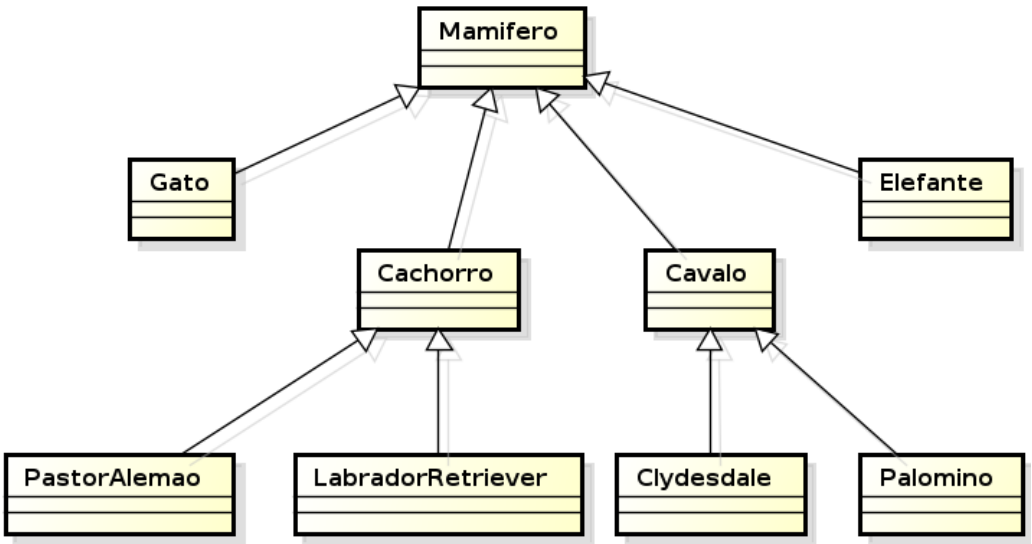


Figura 4.4: Representação da relação de herança. Um *LabradorRetriever* é um *Cachorro*. Tudo que caracteriza um *Mamífero*, caracteriza também o *Gato*, o *Cachorro*, o *Cavalo* e o *Elefante*

Implementação

A linguagem de programação utilizada neste trabalho é *Python* (PYTHON SOFTWARE FOUNDATION, 2012b). *Python* é uma linguagem de programação de alto nível de propósito geral (PYTHON SOFTWARE FOUNDATION, 2012a) com enfâse na facilidade de leitura e codificação. É uma linguagem de fácil prototipação de ideias e algoritmos, principalmente por

sua sintaxe concisa, ser uma linguagem fracamente tipada e possuir uma grande e compreensiva biblioteca padrão.

Baseado nos conceitos de *Java* (ORACLE, 2012), no modelo existe uma classe *Object* é a classe mais genérica do modelo, sendo utilizado como a base de todas as classes. Todo e qualquer objeto deve especializar esta classe. Abaixo segue o protótipo da classe *Object* em *Python*:

```
class Object:

    def __init__(self):
    """ Inicializa os atributos"
        self.__point_cloud_xyz = None
        self.__point_cloud_normals = None
        self.__vfh = None
        self.__bounding_box = None

    def __estimate_normals(self)
    """
    Calcula as normais da nuvem de pontos
    """

    """
    Estima o Viewpoint Feature Histogram da nuvem de pontos
    """
    def __estimate_vfh(self)

    def __estimate_bounding_box(self)
    """
    Estima um Bounding Box que engloba a nuvem de pontos
    """

    def distanceBetween(self, object)
    """
    Calcula a distância deste objeto a um outro objeto passado como parâmetro
    """
```

Duas observações devem ser feitas:

1. Por ser uma linguagem fracamente tipada, em *Python* ao definir um atributo, não é necessário definir o tipo deste atributo;

2. Em *Python* para definir um atributo ou método privado deve-se iniciar o seu nome com dois *underscore* (__). Isto é apenas uma sintaxe da linguagem para identificar que um membro seja considerado privado, porém na prática ainda é possível ter acesso a este membro;

Portanto, o objeto mesa (*Table*) deve herdar de *Object* e sobreescrever todos os métodos.

Nas próximas seções são apresentados em cada um dos passos do *pipeline*. Conforme o modelo, após a aquisição de dados é realizado o *downsampling* (Seção 4.2.2), segmentação (Seção 4.2.3) e anotação (Seção 4.2.4).

4.2.2 *Downsampling*

Algoritmos que trabalham com o processamento de nuvem de pontos tendem a ser custosos, sendo sua complexidade geralmente associado a quantidade de pontos contidos na nuvem. Uma alternativa para reduzir o custo dos algoritmos consiste em realizar um *downsampling* na nuvem de pontos. *Downsampling* consiste na técnica, geralmente utilizada na área processamento de sinal, que busca reduzir a taxa de amostragem de um sinal. Isto pode ser feito de duas maneiras: reduzindo a taxa de dados ou o tamanho do dado propriamente. Como resultado, obtêm-se uma redução significativa na quantidade de dados de uma maneira geral. O algoritmo de *downsampling* é apresentado posteriormente, porém primeiramente é preciso entender o que é uma *kd-tree*, uma vez que é uma estrutura que será bastante utilizada nos algoritmos de manipulação de nuvem de pontos.

kd-tree

k-dtree (abreviação de *k-dimensional tree*, em português árvore *k*-dimensional) é uma estrutura de dados criada por Jon Louis Bentley. Em Ciência da Computação, uma *kd-tree* é uma estrutura de dados de particionamento de espaço que permite organizar pontos em um espaço *k-dimensional*. São um caso especial de árvores de particionamento binário. *Kd-trees* são estruturas de dados úteis para diversas aplicações, tais como busca envolvendo pesquisas em dados multidimensionais (por exemplo, pesquisas de amplitude (*range*) ou dos *k* vizinhos mais pró-

ximo). O objetivo principal da utilização da *kd-tree* é para realizar a busca dos k vizinhos mais próximos.

A *kd-tree* é uma árvore binária na qual todo nó é um ponto em um espaço *k-dimensional*. Todo nó não folha pode ser visto como implicitamente gerando uma divisão no hiperplano que divide o espaço em duas partes, conhecidos como espaços metade (*half-spaces*). Pontos que estão a esquerda do hiperplano são representados pela subárvore esquerda do nó e pontos a direita do hiperplano são representados pela subárvore esquerda. A direção do hiperplano é escolhida da seguinte forma: todo nó na árvore está associado com uma das *k-dimensões*, com o hiperplano perpendicular ao eixo daquela dimensão. Então, por exemplo, se para uma particular divisão o eixo "x" é escolhido, todos os pontos na subárvore que são menores que o valor "x" irão aparecer a subárvore esquerda e todos os pontos maiores que o valor "x" irão aparecer a subárvore direita. Neste caso, o hiperplano iria ser o ponto com o valor "x", e sua normal seria a unidade do eixo x.

O algoritmo para construção da *kd-tree* funciona da seguinte forma: cria-se o "invariante" para qualquer nó, todos os nós que estão na sua subárvore esquerda estão em um lado do plano de divisão, e todos os nós que estão a subárvore direita estão do outro lado. Pontos que estão no plano de divisão podem aparecer em qualquer um dos lados. Suponha por exemplo que seja construída uma *kd-tree* a partir do conjunto de pares (x, y): (2, 3), (5, 4), (9, 6), (4, 7), (8, 1), (7, 2). A Figura 4.5 apresenta a *kd-tree* e a Figura 4.6 apresenta a árvore resultante.

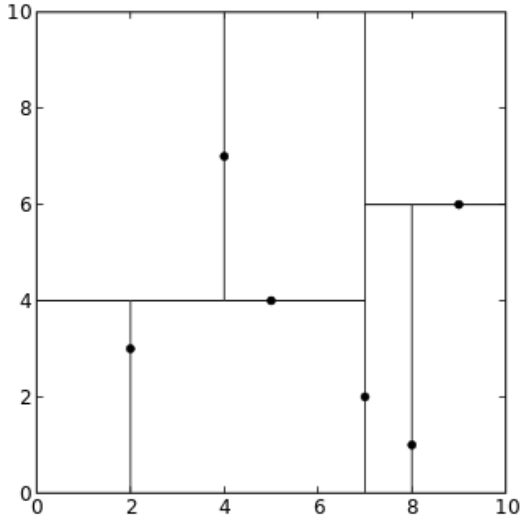


Figura 4.5: Decomposição da *kd-tree* resultante (Wikimedia Foundation, 2012)

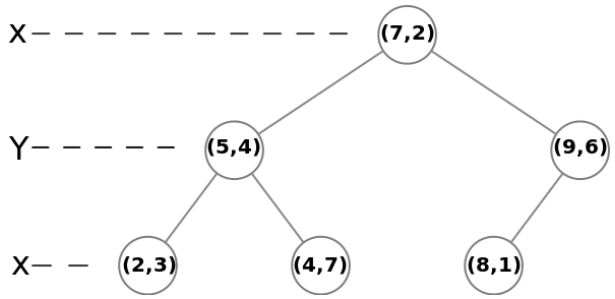


Figura 4.6: Árvore da *kd-tree* resultante (Wikimedia Foundation, 2012)

Algoritmo *Downsampling*

Neste trabalho, foi feito uma redução na quantidade de pontos que representa uma imagem. A ideia consiste em "amostrar" a nuvem de pontos, permitindo que a quantidade de pontos fosse reduzida de forma significante, sem perda significativa de informação.

Algoritmo 5: Algoritmo *Downsampling*

```
1  início
       Entrada: nuvem de pontos P
       Saída: nuvem de pontos Q
2      instanciar KD como uma kd-tree a partir da nuvem de pontos P
3      instanciar Q como uma nuvem de pontos vazia
4      para cada ponto p_i ∈ P faça
5          se p_i não foi processado então
6              marcar p_i como processado
7              adicionar p_i à Q
8              pesquisar em KD pelo conjunto p_i^k de pontos vizinhos de p_i em uma esfera
               com raio r < d
9              para cada ponto p_i^k faça
10                 marcar p_i^k como processado
```

O Algoritmo 5 corresponde ao algoritmo de *downsampling* proposto. A ideia central consiste em receber uma nuvem de pontos *3D* e como resultado gerar uma nuvem de pontos com uma quantidade menor de pontos, porém sem perda significativa de informação. Na linha 2-3 são iniciadas as estruturas utilizadas ao longo do algoritmo: uma *kd-tree* construída a partir de P e uma nuvem de pontos Q que irá ser retornada. Na linha 4 é iniciado um laço onde é percorrido cada um dos pontos em P. Se o ponto p_i já foi processado, prossegue-se para a próxima iteração. Caso contrário, adiciona-se p_i à Q, que correspode à nuvem de pontos que será gerada. Pesquisa-se então por todos os pontos vizinhos de p_i que estejam a uma distância menor que d (um parâmetro também fornecido). Todos os pontos vizinhos retornados são marcados como processados. Ao final de todas as iterações, retorna-se Q como a nuvem de pontos gerada.

Utilizando como métrica de 0.01 unidades de distância, os resultados obtidos foram que, em média, após a aplicação do algoritmo de *downsampling* na nuvem de pontos, esta foi reduzida para aproximadamente 10% de seu tamanho original (considera-se tamanho de uma nuvem de pontos como o número de pontos que a constitui), sem perda aparente de informação, conforme pode ser observado na Figura 4.7.

Na Figura 4.7 são apresentadas duas imagens: o antes e o depois da aplicação do algoritmo de *downsampling* em uma nuvem de pontos. Como observado, não houve perdas significativas de informação após a aplicação do algoritmo.

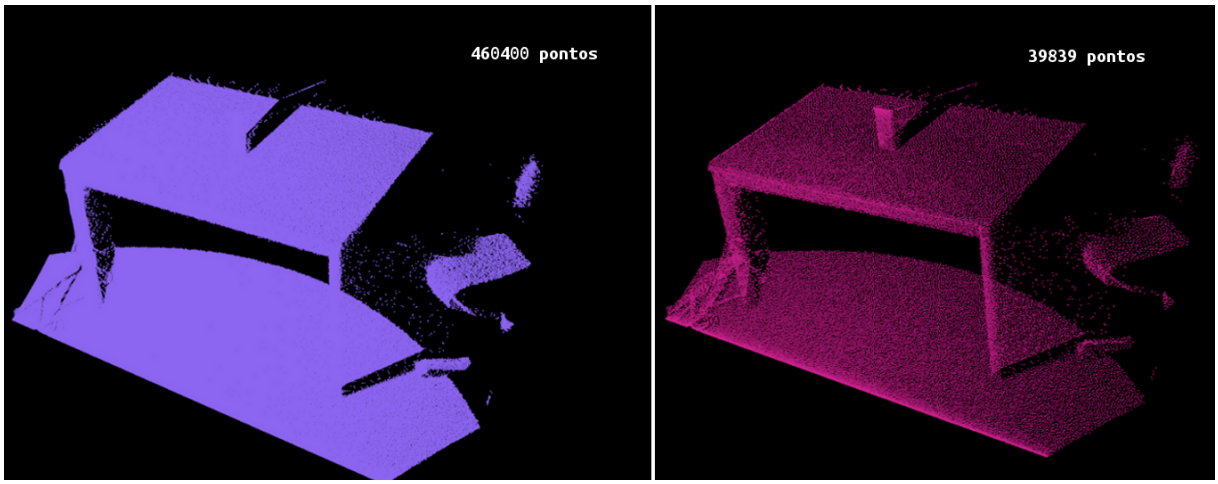


Figura 4.7: Comparação da nuvem de pontos antes e depois a aplicação do algoritmo de *downsampling*. Lado esquerdo e direito, respectivamente antes e depois do *downsampling*

Conforme apresentado Figura 4.7, a nuvem de pontos original (à esquerda) é composta de 460400 pontos. Após a aplicação do algoritmo de *downsampling* a nuvem de pontos resultante (à direita) é constituída de 39839 pontos. Uma redução de mais de 90% no número de pontos.

4.2.3 Segmentação

Sem qualquer pós-processamento, a nuvem de pontos capturada não tem qualquer conceito de objetos. Todos os pontos juntos formam "um objeto só". A técnica de segmentação consiste em "quebrar" a nuvem de pontos em segmentos. Estes segmentos farão sentido quando fornecidos de entrada para a fase de Anotação, no qual são atribuídas classes aos segmentos.

Dentre os diversos tipos de imagens que são encontradas na literatura, as mais comuns são: imagens com intensidade de luz (tradicionais), imagens com informação de profundidade nos *pixels*, imagens de ressonância magnética (também conhecidas como MRI - *Magnetic Resonance Image*) e imagens termais (PAL; PAL, 1993). O foco deste trabalho consiste no estudo

das imagens com informação de profundidade.

Diferentes abordagens de segmentação sugeridas na literatura diferem primariamente no método ou critério utilizado para medir a similaridade entre um conjunto de pontos e então agrupá-los em um segmento. Uma vez que a medida de similaridade esteja definida, segmentos podem ser obtidos agrupando-se aqueles pontos onde a distância de similaridade esteja abaixo de um limiar de corte e que estejam conectados espacialmente (RABBANI, 2006). Dado a natureza de imagens *3D*, estas podem ser tratadas como um conjunto de imagens *2D* (referenciadas como $2\frac{1}{2}D$) permitindo-se que muitas das técnicas de segmentação de imagens *2D* possam ser aplicadas ao problema.

Segundo (RABBANI, 2006), uma técnica de segmentação de imagens se enquadra em uma das três categorias:

1. Segmentação baseada em arestas: Algoritmos deste tipo possuem duas etapas: detectar as arestas que compõe a borda de diferentes regiões, seguido por agrupar os pontos que estejam dentro do limite das arestas, resultando-se em segmentos (WÄHLBY, 2004; XIANG; WANG, 2004; MANGAN; WHITAKER, 1999; SAPPA; DEVY, 2001);

2. Segmentação baseado em superfície: Métodos baseados em superfície utilizam de propriedades locais da superfície como medida de similaridade e agrupam os pontos que estejam espacialmente próximos e que possuem propriedades de superfície similares. Embora sejam menos sensíveis a ruído, geralmente apresentam melhor resultado que algoritmos de detecção de arestas (NAIN, 2006; RUSU, 2008; RABBANI, 2006);

3. Segmentação baseada em *scan-lines*: No caso de imagens com profundidade, cada linha é considerada um *scan-line*, que pode ser tratado independente de outros *scan-lines* no estágio inicial. Esta técnica usa o fato de um *scan-line* em qualquer plano *3D* gera uma linha *3D*. A ideia consiste em detectar os segmentos de linha no primeiro estágio, seguido por agrupar as linhas adjacentes com propriedades similares para formar segmentos planares (VOSSELMAN, 2004; JIANG; BUNKE, 1994);

A proposta deste trabalho consiste em utilizar uma técnica de segmentação baseada em superfície que utiliza métricas de localidade espacial, geometria e coloração. Em alto nível, o algoritmo de segmentação recebe como entrada uma nuvem de pontos e, utilizando diversas métricas, "quebra" a nuvem de pontos em segmentos, gerando diversos arquivos, cada arquivo representando um segmento.

O primeiro passo na segmentação consiste na remoção da maioria do fundo capturado do

ambiente. Isto pode ser feito durante a aquisição da profundidade dos pontos. Normalmente, são eliminados os pontos que estiverem acima de um *threshold* (limiar) de profundidade (por exemplo, não capturar pontos que estiverem acima de 3 metros de distância da câmera). Este passo elimina a maioria dos pontos que estão ao fundo do objeto e que encontram-se distante do mesmo.

Em uma segmentação genérica, normalmente o segundo passo consiste na retirada da superfície em que o objeto mapeado se encontra. Através da aplicação da técnica "*RANSAC plane fitting*" (FISCHLER; BOLLES, 1981) é possível encontrar a superfície plana e com isso considerar apenas os objetos que estiverem acima desta superfície como sendo pertencentes ao objeto. Esta técnica se encaixa na categoria de segmentação por profundidade e apresenta bons resultados, todavia ainda é uma técnica problemática para objetos pequenos, escuros, transparente, reflexivo ou que possuem a mesma coloração que o plano.

A técnica de segmentação implementada neste trabalho aplica-se uma heurística baseada em localidade espacial e geometria. A utilização desta técnica também resolve os problemas apresentados na técnica "*RANSAC plane fitting*".

No Algoritmo 6 é apresentado o passo a passo da execução do algoritmo de segmentação. O algoritmo de segmentação proposto consiste primeiramente da construção de uma *kd-tree* a partir da nuvem de pontos. Uma *kd-tree* permite buscar de maneira eficiente os k vizinhos mais próximos, considerando-se a métrica de distância como a distância limite. A métrica de distância euclidiana define quão perto dois pontos devem estar para que sejam considerados pertencentes a um mesmo segmento ou não. A distância utilizada é de 0.02 unidades. Por meio de testes empíricos, o valor de 0.02 apresentou os melhores resultados. Dois pontos que estejam a mais de 0.02 unidades de distância (e que não tenha nenhum outro ponto entre ambos) não pertencem ao mesmo segmento.

Algoritmo 6: Algoritmo Segmentação 3D

```
1  início
       Entrada: uma nuvem de pontos P
       Saída: uma lista de nuvens de pontos C
2      instancie KD como uma kd-tree a partir de P
3      instancia C como uma lista vazia
4      instancie Q como uma fila vazia
5      para cada ponto p_i ∈ P faça
6          adicionar p_i à Q
7          enquanto existir p_i ∈ Q não processado faça
8              marcar p_i como processado
9              pesquisar em KD pelo conjunto p_i^k de pontos vizinhos de p_i em uma esfera
               com raio r < d
10             para cada vizinho p_i^k de p_i faça
11                 se p_i^k ainda não foi processado então
12                     marcar p_i^k como processado se o ângulo entre os vetores normais de
                       p_i e p_i^k for menor que α então
13                         se a distância da cor entre os dois pontos p_i e p_i^K for menor que
                           β então
14                             adicionar p_i^k à Q
15         se Q possui pelo menos 100 pontos então
16             adicionar Q à lista de segmentos C
17         reiniciar Q como uma fila vazia
```

A parte central do algoritmo de segmentação consiste na heurística de localidade espacial, geometria e coloração. A informação de localidade espacial é checada na linha 9, onde são considerados apenas pontos que estão a uma distância menor que r (0.02 unidades) de um ponto. Na linha 12, a métrica de geometria calcula o ângulo formado entre o vetor normal de dois pontos. Se o ângulo calculado for menor que um ângulo de corte α, significa que estes pontos são pertencentes ao mesmo segmento. Baseado em testes empíricos, o valor do ângulo de corte α de 26 radianos foi o que apresentou melhores resultados.

Na linha 13, a métrica de cor calcula a distância entre as duas cores que representam os pontos. Se a distância das cores (também conhecido como ΔE na ciência de cores) obter um resultado maior que um valor β, neste caso 10, então significa que os dois pontos possuem uma

discrepância muito grande de cores e indica que pertencem a segmentos distintos.

Em estudo das cores, a "distância" entre duas cores é referenciada como ΔE. ΔE surgiu depois do *CIE* (*Commission Internationale De L'Eclairage of International* ou *Comission on Illumination*) anunciar em 1976 dois novos modelos de aparência de cores (*L*a*b** e *L*u*v**) com o objetivo de substituir o modelo padrão *XYZ* utilizado desde 1931 para mensurar a distância entre cores. A razão da adoção do novo padrão surgiu do fato de pesquisadores já terem conhecimento a algum tempo que o padrão existente não era perceptualmente uniforme, ou seja, o modelo de cores não modelava de maneira precisa o que seres humanos observavam. O olho humano é mais sensível em algumas áreas de cores e menos sensível em outras, um fato que inicialmente a fórmula não levava em consideração. A Tabela 4.5 apresenta a relação de alguns intervalos de valores do ΔE e seus significados.

Tabela 4.5: ΔE e suas variações. Como apresentado, ΔE menores que 1 não são distinguíveis por seres humanos

Valor ΔE	**Significado**
0-1	Diferença normalmente invisível
1-2	Uma pequena diferença, apenas óbvia para um olho treinado
2-3.5	Diferença média, óbvia também para um olho não treinado
3.5-5	Uma diferença óbvia
> 6	Uma diferença muito óbvia

O espaço de cor *L*a*b* (ou somente *Lab*) calcula a distância euclidiana, ou seja, simplesmente calcula a distância entre dois pontos em um espaço tridimensional de cores. Como os valores determinados são baseados em fórmulas matemáticas, é importante que seja levado em consideração o modelo de cor que se está utilizando, ou seja, aplicar a distância euclidiana em diferentes modelos irá produzir resultados distintos.

Mas por quê não utilizar distância com duas cores representadas no *RGB*? *RGB* é um sistema não linear. Muitas vezes seleciona cores muito escuras ou muito azuladas. *YUV* é sempre melhor que *RGB*, entretanto está distante do ideal. *CIEL L*u*v* funciona bem para a maioria dos casos, porém em algumas situações produz erros inaceitáveis. A versão modificada de *CIEL L*u*v** de (GRANGER, 1994) executa melhor. Entretanto, mesmo com a curva de luz modificada, *L*u*v** não funciona muito bem para cores rosas (por exemplo, a cor da pele de pessoas caucasianas) (RIEMERSM, 2012).

Esta é a motivação para utilizar medida de distância nos modelos *CIE Luv e CIE Lab* (ou simplesmente, *CIE Luv* e *CIE Lab*). Estes são modelos considerados perceptualmente uniformes, ou seja, duas cores que estão distantes no espaço de cor são distantes perceptualmente. Originalmente, a ideia era que *CIE Lab* e *CIE Luv* fossem diferentes, mas de certa forma equi-

valentes em como representam um espaço uniforme de cores, ou seja, muito melhor que outras representações como *RGB*, *CIE XYZ*, e outros mas muitos longe do perfeito. *CIE Luv* tem a vantagem de ser uma transformação mais linear em um plano matiz a partir de *CIE XYZ* do que *CIELAB*, e então favorece coisas como diagramas de cromacidade perceptual. Entretanto, toda a atividade em melhorar métricas de cores tais como o *CIE94* e o *CIE2000* são baseados no *CIE Lab*. Aparentemente, *CIE94* e *CIE2000* se apresentam como uma métrica mais adequada para situações de cores reflexivas e com a atualização em 2000, *CIE Lab* se tornou uma métrica bastante adequada para todas as situações. O ΔE utilizado nos experimentos calculado com duas cores representadas no modelo *CIE Lab*.

Antes disso, é necessário que sejam estimados os vetores normais dos pontos da nuvem. Isto é feito por uma chamada externa ao algoritmo "*Estimating Surface Normals in a Point Cloud*" presente na PCL (*Point Cloud Library*) (POINTCLOUD.ORG, 2012). Se a heurística retornar verdadeiro então estes pontos são considerados pertencentes a um mesmo segmento.

Na Figura 4.8 é exibido o resultado do algoritmo de segmentação. O *dataset* de imagens tridimensionais utilizados neste trabalho foi retirado de (LAI, 2011a). Dado como entrada uma nuvem de pontos, o algoritmo extrai segmentos. Na imagem acima é apresentada a nuvem de pontos antes da segmentação, abaixo a nuvem de pontos após a segmentação. Note que a figura abaixo é composta de nuvem de pontos menores cada uma com uma cor. Cada cor representa um segmento e para cada segmento o programa gera um arquivo contendo a nuvem de pontos que representa esse segmento.

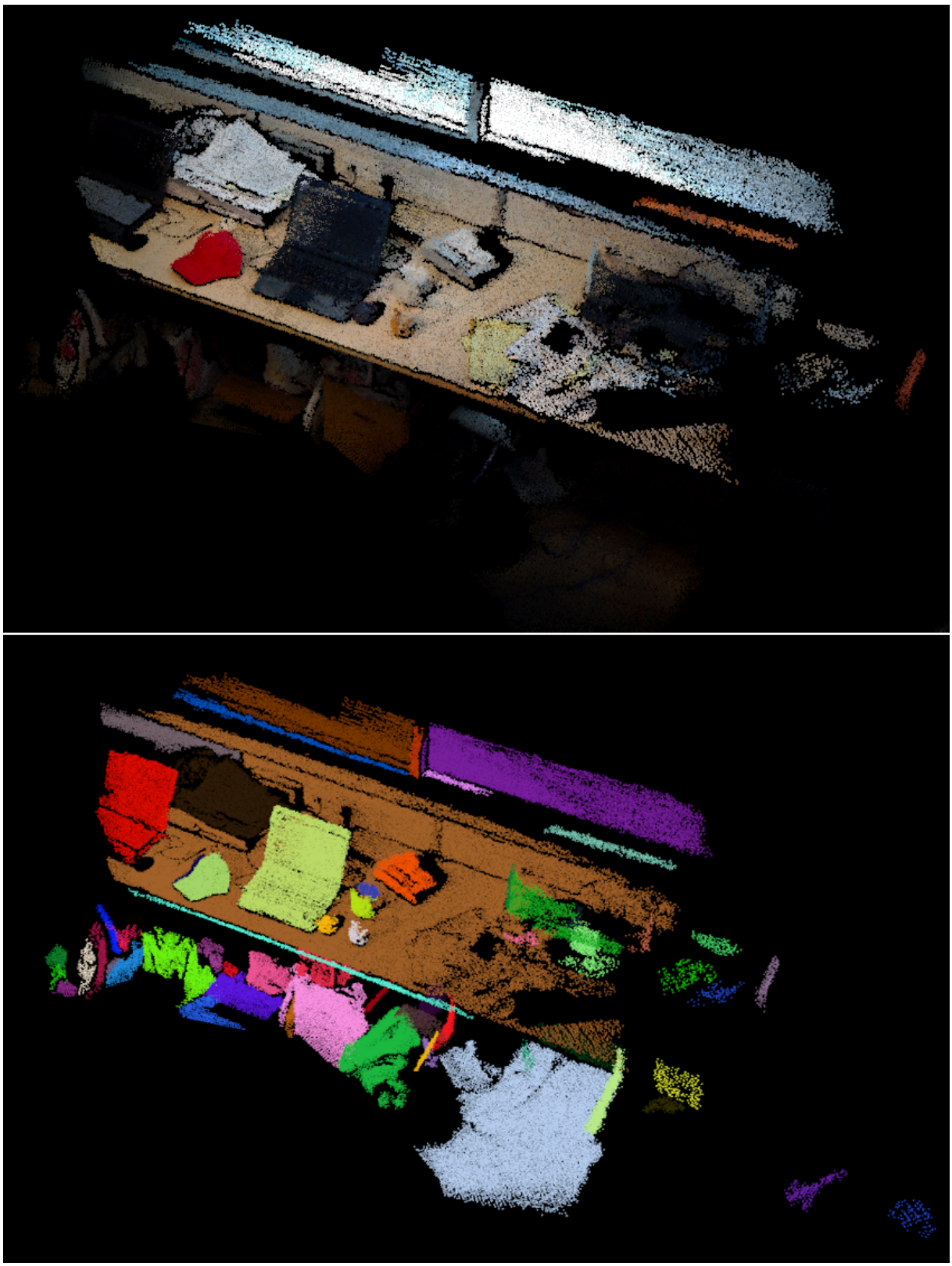


Figura 4.8: Resultado da nuvem de pontos após a segmentação. Acima é exibida a nuvem de pontos original, onde não existe o conceito de segmentos. Abaixo a nuvem de pontos resultante do algoritmo. Cada cor representa um segmento

Na composição do segmento foi definido que este deve ser formado por no mínimo 100 e no

máximo 25000 pontos. Segmentos com menos de 100 pontos podem ser considerados ruídos. Já o limite de 25000 é suficientemente grande, visto que nesta nuvem de pontos fornecida como entrada já foi aplicada a técnica de *downsampling* (esta técnica apenas tem por objetivo reduzir o número de pontos que representa o objeto para reduzir o tempo dos algoritmos que processam nuvem de pontos);

Experimentos e Resultados

Para evitar cálculos complexos que envolvem a transformação dos modelos de cores RGB e Lab, foi utilizado a biblioteca *Python-Colormath* (SNAGGLEPANTS, 2012).

Os experimentos foram realizados da seguinte forma: dado uma nuvem de pontos fornecida como entrada (imagens tridimensionais do *dataset* utilizado neste estudo foram retiradas de (LAI, 2011a)), é aplicado o algoritmo de segmentação proposto sem e com a utilização de informação de cor e comparado como *benchmark* o algoritmo contido na PCL. As imagens estão dispostas da seguinte forma: no canto esquerdo superior está a imagem original, no canto direito superior está o resultado da segmentação utilizando o algoritmo da PCL, no canto inferior esquerdo e direito estão o resultado do algoritmo de segmentação proposto respectivamente sem e com a utilização de informação de cor.

Na Figura 4.9 é mostrada a nuvem de pontos de uma mesa em um escritório. Sobre a mesa estão dispostos diversos objetos, sendo os mais visivelmente identificáveis: boné, *notebook*, *mouse*, lata de refrigerante, caneca e algumas folhas de papéis. Sem a utilização da métrica de cor, a imagem está particionada em 40 segmentos, entretanto alguns objetos sobre a mesa não foram segmentados corretamente, entre os mais notáveis: *mouse*, *notebook* e boné. No canto inferior direito da imagem é apresentado o resultado da técnica de segmentação considerando métrica de cor. Foram gerados 43 segmentos, apresentando uma melhora que pode ser notada visualmente pela figura. O *mouse*, *notebook* e boné foram segmentados corretamente. O algoritmo implementado na PCL não conseguiu segmentar objetos tais como o *notebook* e o boné.

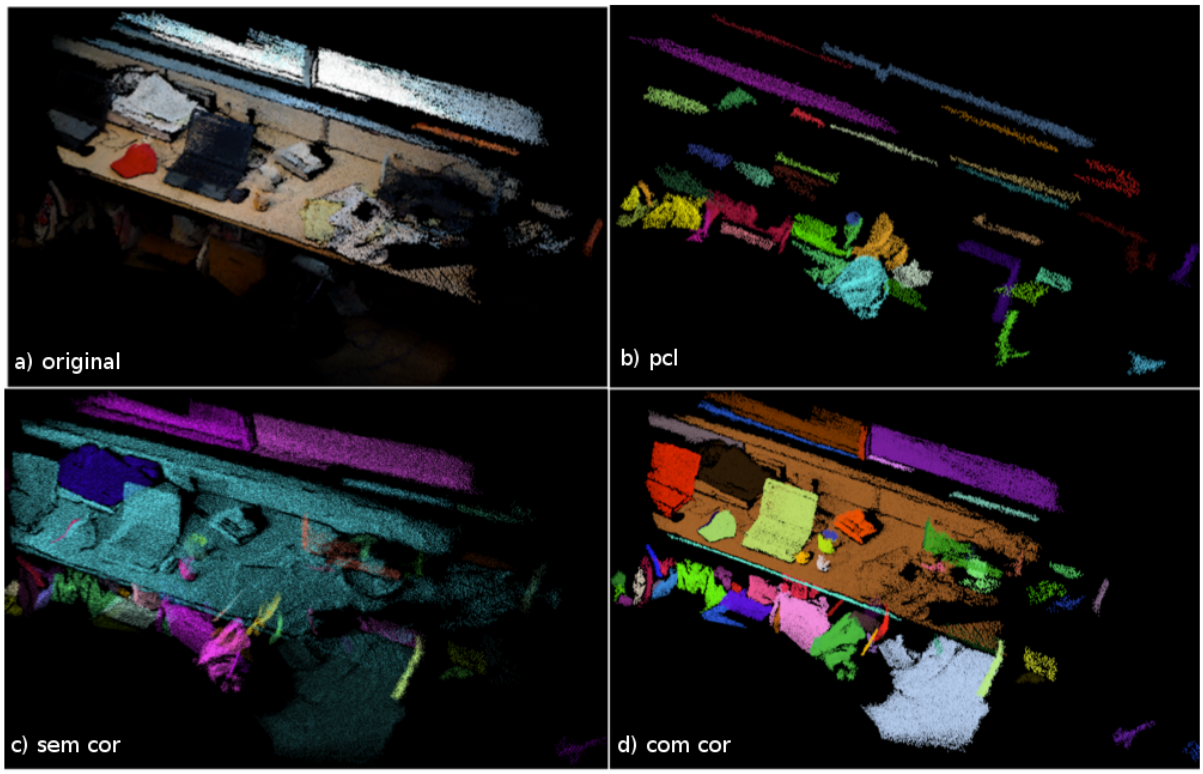


Figura 4.9: Nuvem de pontos de uma mesa em um escritório. b) Algoritmo de segmentação da PCL: 38 segmentos gerados. c) Algoritmo de segmentação sem utilizar métrica de cor: 40 segmentos gerados gerados. d) Algoritmo de segmentação proposto utilizando métrica de cor: 43 segmentos gerados.

Na Figura 4.10 é fornecida uma nuvem de pontos de uma pequena porção de uma cozinha. Sobre a mesa estão dispostos vários pequenos objetos, inclusive alguns parcialmente oclusos. Em uma situação ideal, todos os objetos deveriam ser mapeados corretamente. Entretanto, embora longe do ideal, pode ser observado que a segmentação de imagens utilizando métricas de cores (canto inferior direito) apresentou melhores resultados. Enquanto a segmentação sem a utilização de métricas de cores gerou 57 segmentos de cores, segmentação utilizando a métrica de cor gerou 64 segmentos. Visualmente pode ser observado que objetos como a pilha de pratos, lanterna e a caneca foram segmentados corretamente no algoritmo de segmentação proposto com a utilização da informação de cor. Provavelmente devido a ruídos, oclusão parcial de objetos e *outliers* acabou levando a erros na segmentação. Mesmo a PCL apresentando melhora no desempenho comparado com o *dataset* anterior, ainda assim apresentou resultados piores que o algoritmo proposto.

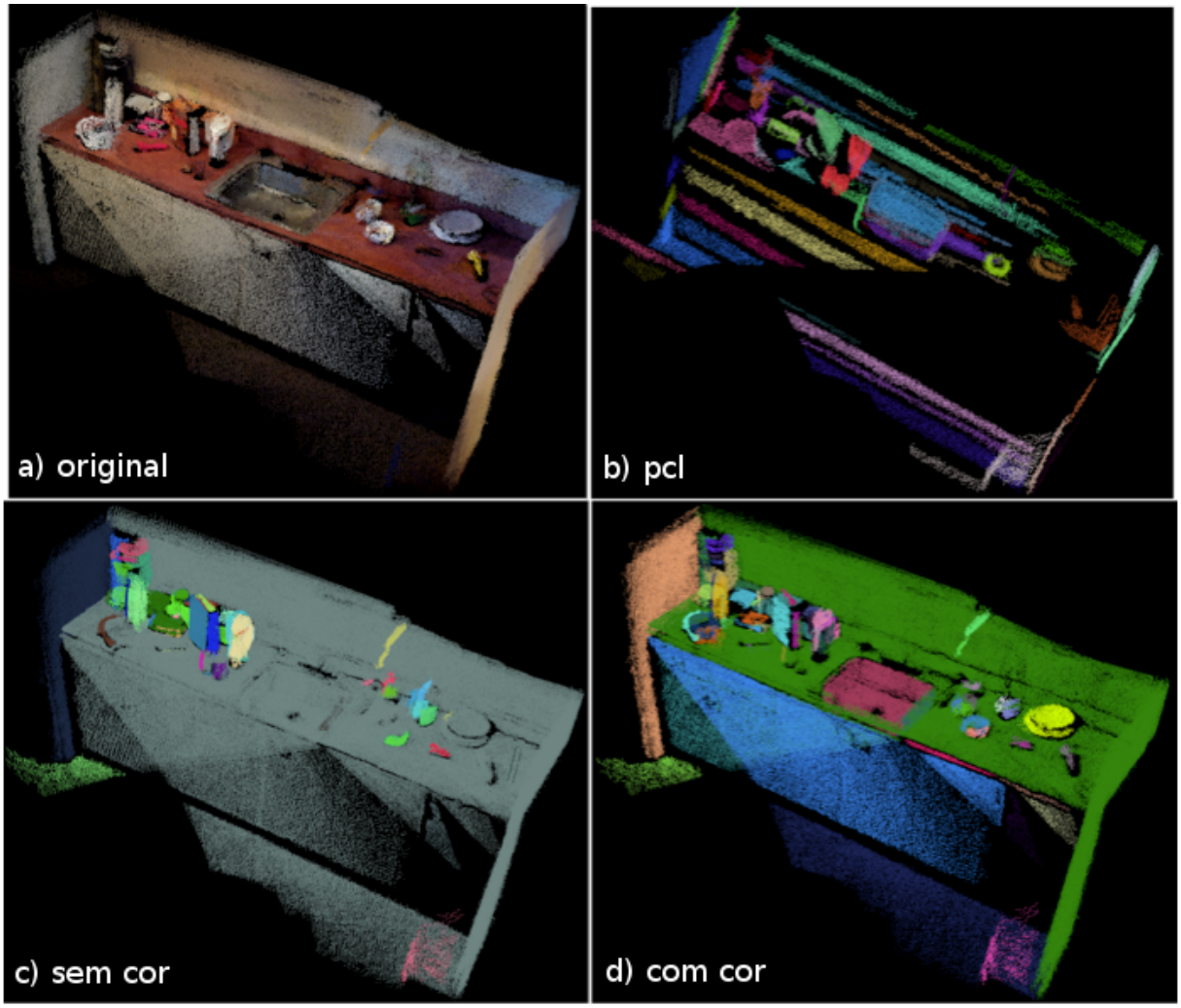


Figura 4.10: Nuvem de pontos do mapeamento de uma pequena porção de uma cozinha. b) Algoritmo de segmentação da PCL por sua vez gerou 63 segmentos. c) Segmentação proposta sem a utilização de informação de cor gerou 57 segmentos. d) Algoritmo proposto utilizando a informação de cor 64 segmentos.

Novamente, como mostrado na Figura 4.11, embora a segmentação não tenha sido ideal em ambos os casos, a utilização do algoritmo proposto com a métrica de cor apresentou o melhor resultados. Ambos o boné e a tigela foram segmentados corretamente apenas no algoritmo proposto. Neste caso, o algoritmo da PCL apresentou resultados próximos, porém ainda sim piores.

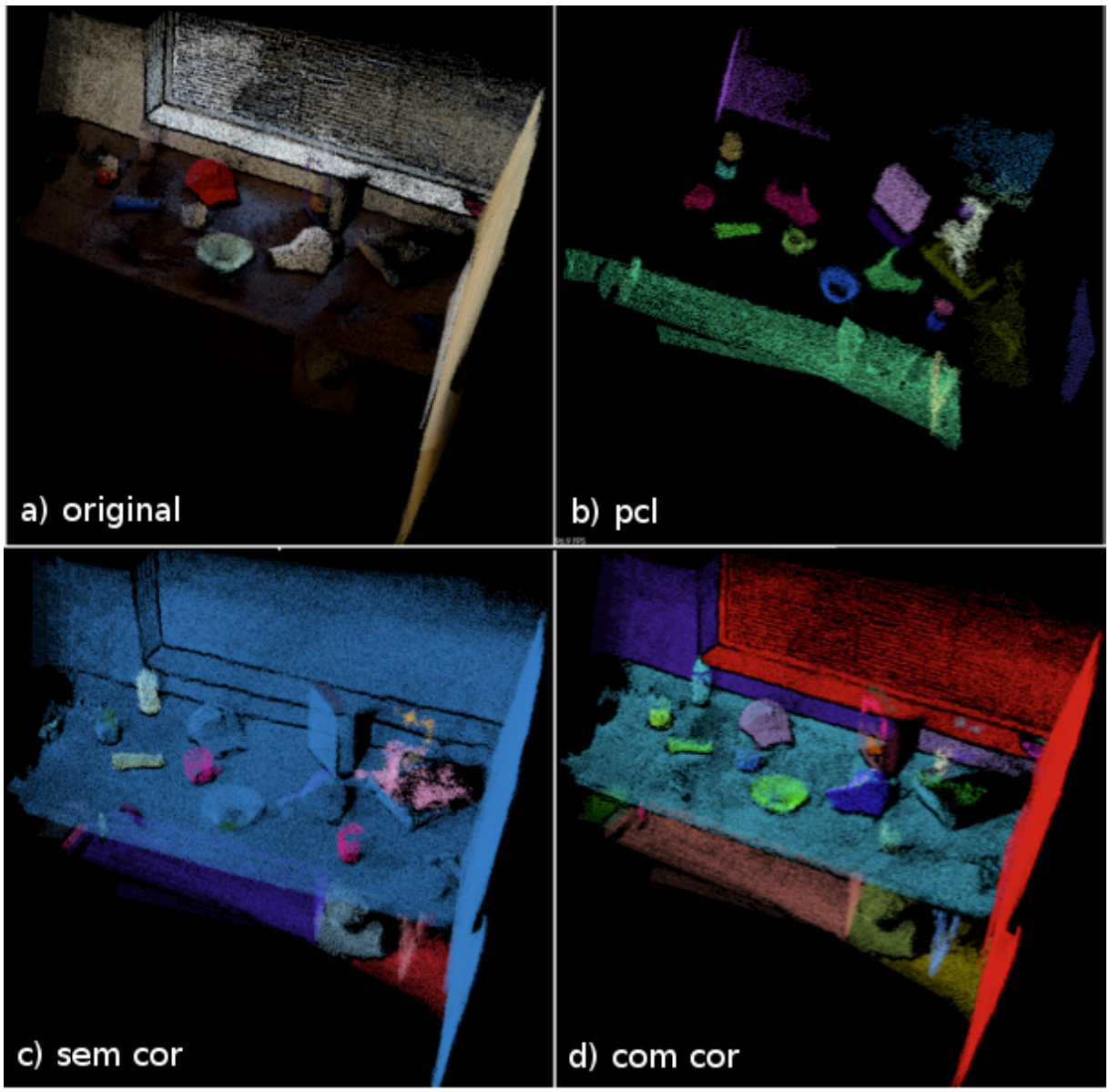


Figura 4.11: Nuvem de pontos do mapeamento de uma mesa. b) Algoritmo de segmentação da PCL: 20 segmentos gerados. c) Algoritmo de segmentação proposto sem utilizar a informação de cor: 23 segmentos gerados. d) Algoritmo proposto utilizando informação de cor: 37 segmentos gerados.

Por meio dos experimentos e resultados gerados pode ser observado que o algoritmo de segmentação proposto apresentou resultados melhores que o contido na biblioteca PCL. A utilização da métrica de cor mostrou ser bastante interessante de ser incluída em uma segmentação baseado em superfície, provendo bons resultados. Se geometricamente pode não existir uma diferença significativa entre dois objetos, a informação de cor pode ser a solução. Além disso, os resultados mostram que o algoritmo proposto apresentou melhores resultados (como pode ser observado visualmente nas figuras) que o algoritmo *Euclidean Cluster Extraction* presente na PCL.

Com a nuvem de pontos segmentada, o próximo passo consiste em anotar os segmentos, ou seja, classificar os segmentos. Na próxima seção são apresentados os conceitos de anotação juntamente com as técnicas mais utilizadas.

4.2.4 Anotação

Tradicionalmente, a comunidade de visão computacional tem anotado sequências de vídeos um *frame* por vez. Uma pessoa deve tediosamente realizar a segmentação de objetos utilizando ferramentas de anotação como *LabelMe* (RUSSELL, 2008) e *vatic* (VONDRICK, 2010) (LAI, 2011a).

(LAI, 2011a) propõe uma forma alternativa. Ao invés de anotar cada quadro do vídeo, (LAI, 2011a) primeiramente cria uma reconstrução *3D* da cena. Por fim, os objetos nesta reconstrução *3D* são anotados a mão.

Neste trabalho porém, a maneira utilizada é ainda mais simples. Uma vez que a nossa nuvem de pontos foi segmentada (conforme apresentado na Seção 4.2.3), a princípio os objetos já foram separados dos outros objetos e do ambiente. Basta então manualmente anotar os segmentos importantes e ignorar aqueles que não é de interesse.

Veja ainda que durante a segmentação um objeto pode ser segmentado em objetos menores. Conforme apresentado na Figura 4.12, a nuvem de pontos foi segmentada em 7 componentes. Os objetos são: 1 é uma caixa, 2 é uma superfície de uma mesa, 3 é o pé de uma mesa, 4 é a superfície de um chão, 5, 6 e 7 são *outliers* (partes de objeto, pontos pegados fora do escopo, etc). É interessante notar que juntos os objetos 2 e 3 formam uma mesa.

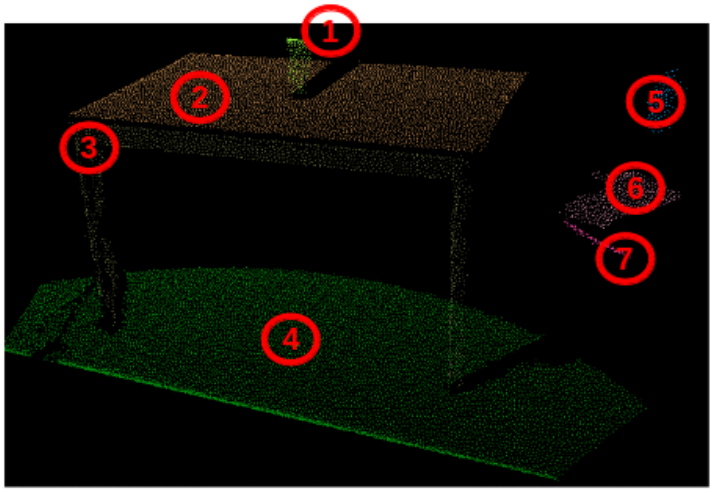


Figura 4.12: Representação de uma possível anotação em uma nuvem de pontos já segmentada. Os únicos segmentos que importantes desta nuvem são: 1, uma caixa, 2 uma superfície de uma mesa, 3 um pé de uma mesa e juntos 2 e 3 formam uma mesa. Os outros segmentos podem ser ignorados na anotação

4.2.5 Extração de Características

Extração de características corresponde a quaisquer características passíveis de extração a partir de uma nuvem de pontos. A obtenção destas características permite, por exemplo, caracterizar uma nuvem de pontos. Estas características são informações relevantes, podendo serem utilizadas em conjunto com alguma fase do *pipeline* de reconhecimento do objeto como: *downsampling*, segmentação ou classificação. Neste capítulo são apresentadas algumas das características obtidas a partir de uma nuvem de pontos.

Na Seção 4.2.5 é apresentado uma técnica que permite estimar os vetores normais dos pontos de uma nuvem. Na Seção 4.2.5 é apresentado sobre o *VFH* (*Viewpoint Feature Histogram*), uma característica capaz de gerar histograma único (como uma assinatura) a partir de informações do objeto.

Estimativa dos Vetores Normais dos Pontos

Dado uma superfície geométrica, é relativamente trivial inferir a direção da normal de um ponto da superfície como um vetor perpendicular à superfície naquele ponto. Nuvem de pontos entretanto são apenas uma amostragem dos pontos que representam a superfície real. Segundo (RUSU, 2009), existem basicamente duas abordagens para estimar o vetor normal dos pontos a partir de uma nuvem de pontos:

- obter a superfície a partir da nuvem de pontos, utilizando técnicas de *meshing* e então

computar os vetores normais da superfície;

- utilizar aproximações para inferir as normais da superfície diretamente na nuvem de pontos;

A segunda abordagem é a utilizada por (RUSU, 2009) e também a que será utilizada neste trabalho.

Embora existam diversos métodos para estimar vetores normais, este é um dos meios mais simples de se fazê-lo (RUSU, 2009). O problema de determinar a normal de um ponto em uma superfície pode ser aproximado ao problema de estimar a normal de um plano tangente à superfície, transformando-o em um problema de estimativa denominado "*least-square plane fitting*".

A solução para a estimativa da superfície normal é então reduzida a análise de *eigenvectors* (autovetores) e *eigenvalues* (autovalores), ou PCA - *Principal Component Analysis*, em português Análise de Componente Principal, de uma matriz de covariância criada a partir dos vizinhos mais próximos.

Na Figura 4.13 é apresentada uma nuvem de pontos representando uma cozinha. As flexas em azul indicam as normais estimadas dos pontos utilizando o método do PCA. Em geral, uma vez que não existe meio matemático para resolver o sinal da normal, sua orientação computada via PCA não é consistente sobre toda a nuvem de pontos. Conforme observado, existem normais apontando em todas as direções, muito embora isto esteja errôneo.

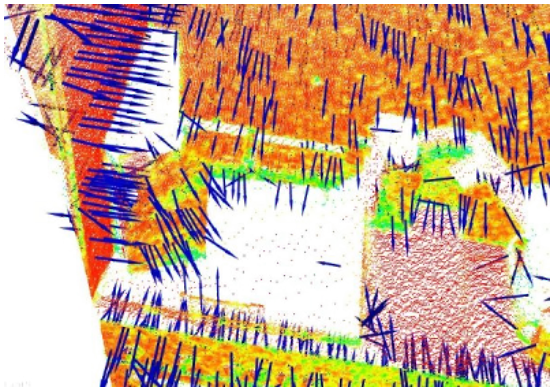


Figura 4.13: Representação de uma parte de uma cozinha. As flexas em azul representam a orientação da normal estimada daquele ponto (RUSU, 2009)

Extended Gaussian Image (EGI) é um recurso utilizado para melhor visualizar a distribuição das normais em uma imagem. Funciona da seguinte maneira: imagine transladar todas as normais para o eixo (0,0,0). Se somente a translação for feita, a direção da normal irá se manter.

Ao fazer isso com todos os vetores normais será obtido uma melhor visualização de como estão distribuída as direções das normais.

Na Figura 4.14 é apresentado o *Extended Gaussian Image* (EGI), também conhecido como esfera normal, utilizada para descrever a orientação de todas as normais de uma nuvem de pontos. Note que a orientação das normais forma uma esfera de 360º, o que está errado. Uma pessoa ao visualizar um objeto só será possível observar normais que apontam na direção de uma semi-esfera.

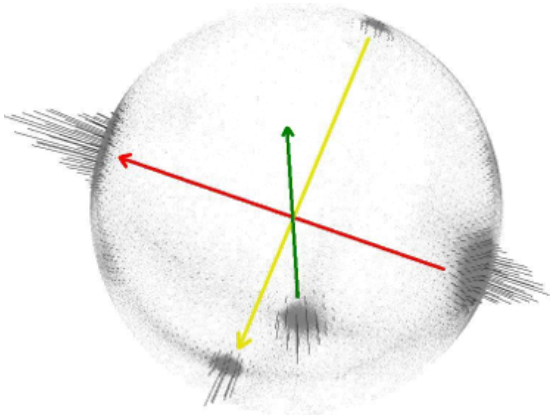


Figura 4.14: *Extended Gaussian Image* (EGI), também conhecido como esfera normal, descreve a orientação de todas as normais de uma nuvem de pontos. Note que a orientação das normais forma uma esfera de 360º, o que está errado (RUSU, 2009)

Na Figura 4.15 são mostrados as possíveis direções de uma normal de um ponto a partir de um ponto de vista. Conforme observado, as direções formam um semiesfera.

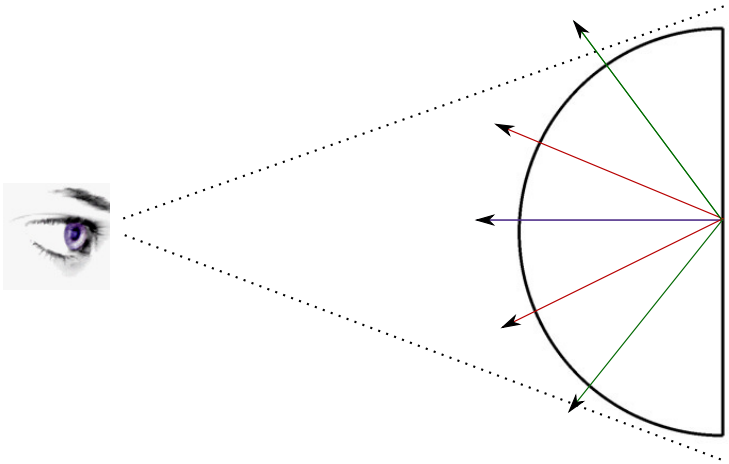


Figura 4.15: Dado um ponto de vista, as possíveis normais corretas dos pontos devem formar uma semiesfera. Se a direção da normal apontar para a parte inexistente da esfera é como se estivesse vendo a parte de trás do objeto (o que é impossível porque o que você vê sempre é a parte da frente)

Se o ponto de vista (*viewpoint*) V_p é de fato conhecido, a solução para este problema é trivial. Para orientar todos os vetores normais $\vec{n}_i$ consistentemente pelo ponto de vista, eles precisam satisfazer a equação:

$$\vec{n}_i.(v_p - p_i) > 0 \tag{4.1}$$

As Figuras 4.16 e 4.17 apresentam o resultado após todas as normais após serem orientadas consistentemente de acordo com o ponto de vista.

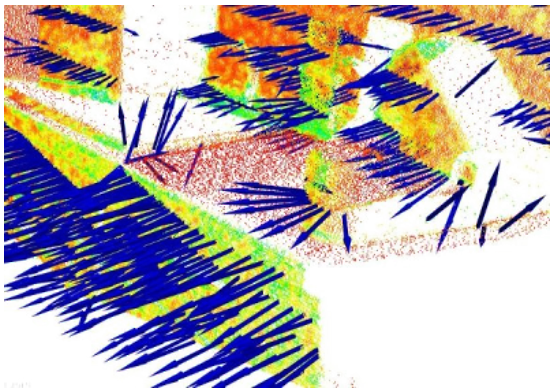


Figura 4.16: Representação de uma parte de uma cozinha após as normais serem orientadas consistentemente de acordo com o ponto de vista (RUSU, 2009)

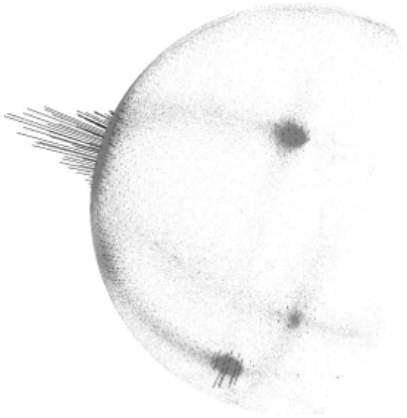


Figura 4.17: *Extended Gaussian Image* (EGI), também conhecido como esfera normal, descreve a orientação de todas as normais de uma nuvem de pontos após as normais serem orientadas consistentemente de acordo com o ponto de vista (RUSU, 2009)

O método apresentado pela equação 4.1 funciona apenas se a nuvem de pontos possui apenas um ponto de vista. Se a base de dados foi adquirida a partir de múltiplos ponto de vistas, então este método de orientação não funciona. (RUSU, 2009) apresenta um algoritmo mais complexo para tratar este caso em especial.

Viewpoint Feature Histogram

Viewpoint Feature Histogram (VFH) é um descritor para nuvem de pontos 3D que codifica geometria e ponto de vista (*viewpoint*). Um descritor é uma "caraterística". É o que identifica e caracteriza algo.

Criado e apresentado por (RUSU, 2010), VFH é dito robusto para operar em superfícies com grandes ruídos e com informações de profundidades ausentes, permitindo trabalhar de forma confiável também com dados obtidos a partir de câmeras *stereo*.

VFH têm suas raízes em *Point Feature Histogram* (PFH) (RUSU, 2008). PFH é um descritor que gera como resultado um histograma que coleciona par a par os ângulos *pan*, *tilt* e *yaw* entre os pares de normais na superfície. Para cada par de pontos $\langle p_i, p_j \rangle$ e sua normal da superfície estimada $\langle n_i, n_j \rangle$, o conjunto desvio angular normal pode ser estimado como:

$$\alpha = v \cdot n_j \tag{4.2}$$

$$\phi = u \cdot \frac{(p_j - p_i)}{d} \tag{4.3}$$

$$\theta = arctan(w \cdot n_j, u \cdot n_j) \tag{4.4}$$

Onde:

$$u = n_i \tag{4.5}$$

$$v = (p_j - p_i) \times u \tag{4.6}$$

$$w = u \times v \tag{4.7}$$

e d é a distância Euclidiana entre os pontos p_i e p_j.

A Figura 4.18 representa graficamente as três características angulares. Uma vez que todos os pares de pontos são considerados, a complexidade computacional de PFH é $O(n^2)$ no número de normais da superfície n.

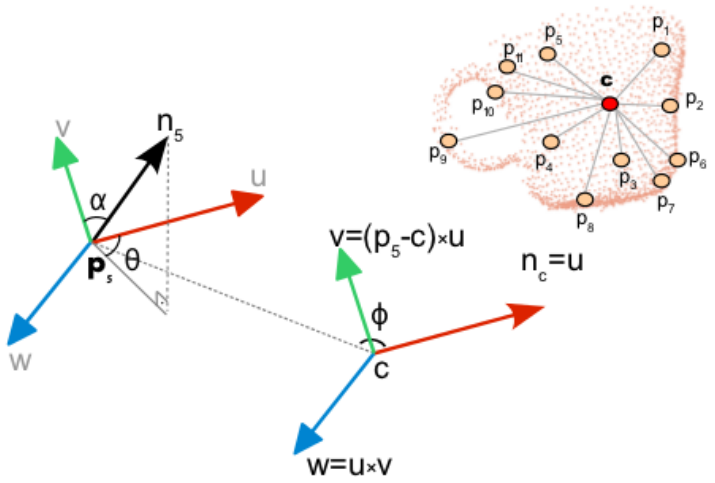


Figura 4.18: *Point Feature Histogram* coleta estatística dos ângulos relativos entre as normais da superfície de cada ponto com a normal da superfície no centróide do objeto. A parte inferior esquerda da figura descreve as três características angulares para um par de pontos dados com exemplo (RUSU, 2010)

Dado um conjunto de pontos $P = i$ com $i = 1...n$, *PFH* captura a quádrupla $\langle \alpha, \phi, \theta, d \rangle$ entre todos os pares (p_i, p_j) de P e modela o resultado como um histograma. O processo de armazenar em um histograma consiste em dividir cada intervalo de possíveis valores de uma característica em b subdivisões, ou seja, dividir cada intervalo de α, ϕ, θ e d em b subdivisões. Uma vez que três das quatro características apresentadas acima (α, ϕ e θ) são medidas entre as normais de um ângulo, seus valores podem ser facilmente normalizados para o mesmo intervalo de um circulo trigonométrico.

Uma proposta para tornar o algoritmo mais eficiente foi desenvolvida, denominada de *Fast Point Feature Histogram* (FPFH) (RUSU, 2009). Assim como PFH, FPFH também estima as mesmas características angulares, porém a estimativa é feita apenas entre os pontos e seus k vizinhos mais próximos, seguido de uma atualização do peso do histograma resultante do ponto com o histograma dos vizinhos, assim reduzindo a complexidade computacional para $O(k*n)$.

VFH consiste em uma melhora no resultado de reconhecimento do FPFH (RUSU, 2010), através da adição de uma variância de ponto de vista, enquanto mantêm a escala invariante. FPFH foi extendido de forma a permitir ser estimado para um *cluster* inteiro de objetos, além de computar estatísticas adicionais entre a direção do ponto de vista e as normais estimadas de cada ponto. Para fazer isso, (RUSU, 2010) utilizou a ideia de adicionar a direção do ponto de vista no cálculo do ângulo normal relativo de FPFH. A complexidade de VFH é $O(n)$.

A Figura 4.20 apresenta o histograma gerado pelo VFH, com a nova característica consistindo de duas partes: (1) *Viewpoint Direction Component* (Figura 4.19) e (2) um componente

da superfície composta de um FPFH estendido (Figura 4.18).

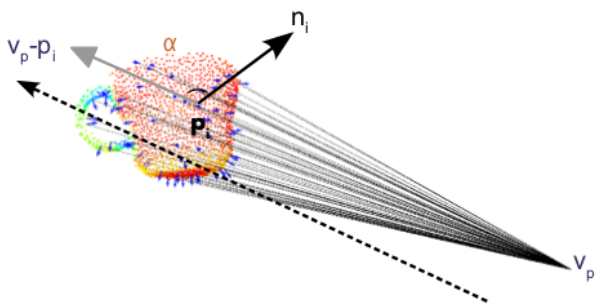


Figura 4.19: *Viewpoint Feature Histogram* é criado a partir de uma extensão do *Fast Point Feature Histogram*, juntamente com as estatísticas dos ângulos relativos entre cada normal da superfície para o direção do ponto de vista central (*central viewpoint direction*) (RUSU, 2010)

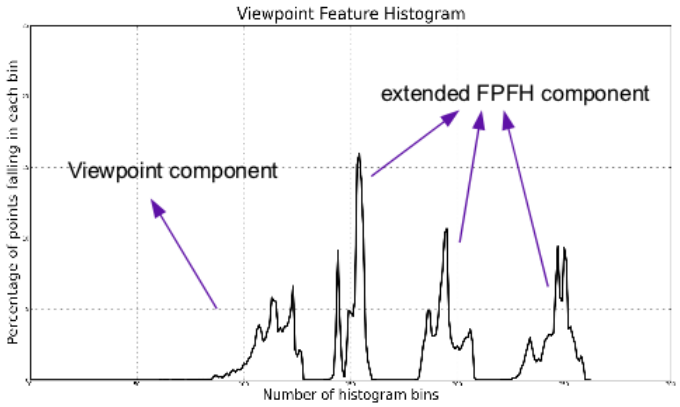


Figura 4.20: Um exemplo de *Viewpoint Feature Histogram* para um objeto utilizado. Note os dois componentes concatenados (RUSU, 2010)

O primeiro componente (*viewpoint component*, componente do ponto de vista) é computado coletando-se o histograma dos ângulos que a direção do ponto de vista faz com cada normal. Note que isto não quer dizer o ângulo do ponto de vista para cada normal, uma vez que isto não seria invariante na escala, mas ao contrário medir o ângulo com o direção do ponto de vista central traduzida em cada normal. O segundo componente (*extended FPFH component*, componente FPFH extendido) mede os ângulos relativos *pan*, *tilt* e *yaw*, mas agora medido entre a direção do ponto de vista no ponto central e cada uma das normais na superfície.

Segundo os testes realizados por (RUSU, 2010), a taxa de acerto do reconhecimento de objeto é de aproximadamente 98.5%. Os testes foram realizados sobre um *dataset* contendo aproximadamente 60 objetos, descritos por cerca de 54000 cenas. Além disso, o algoritmo mostrou-se rápido (cerca de 0.3ms em média) e apto a trabalhar com dados com ruídos. A comparação é feita pelo algoritmo KNN por meio da estrutura kd-tree. A construção da árvore

e a pesquisa pelos vizinhos mais próximos utiliza um peso igual em cada bin do histograma VFH.

No estágio atual de desenvolvimento do *pipeline*, todos os passos apresentados estão implementados e funcionamento conforme apresentado ao longo do trabalho. A continuação do trabalho, como a adição da implementação de novos descritores de nuvem de pontos e também da avaliação do desempenho do reconhecimento ficam a cargo de trabalhos futuros.

5 *Conclusões*

Com o advento e posteriormente popularização de sistemas *RGB-D*, tornou-se possível a obtenção da profundidade de cada *pixel* de uma imagem de forma rápida e confiável. A imagem capturada, agora com *pixels* em um plano tridimensional, ao contrário de imagens comuns que organizam seus *pixels* em um plano bidimensional, é então muito mais rica em informação do que a obtida por câmeras tradicionais. Esta informação adicional possibilita a realização de diversas operações sobre a agora nuvem de pontos. Anteriormente esse tipo de imagem já era possível de ser obtida utilizando a técnica de visão estereoscópica. Porém, como apresentado no trabalho, embora as imagens foram relativamente promissoras utilizando a técnica proposta para melhor emparelhamento de imagens, ainda é muito inferior às obtidas pelo *Kinect*.

A representação tridimensional pode ser considerada muito mais rica quando comparada a representações bidimensionais de dados. O trabalho propõe um estudo sobre a representação e reconhecimento de padrões em estruturas tridimensionais. É apresentado em paralelo um trabalho realizado em reconhecimento de padrões tridimensionais utilizando o *wiimote* (controle do *Wii*), onde captura-se a cada período de tempo estipulado as informações do acelerômetro como uma tripla (x, y, z). O objetivo é conseguir reconhecer movimentos. Conforme apresentado, utilizando-se as técnicas de *KNN*, *TRKNN* e *DTW* em conjunto, foram obtidos excelentes resultados (na grande maioria próximos de 100% de taxa de acerto).

Com respeito ao *Kinect*, o trabalho propõe a captura dos dados por meio do modelo de captura proposto. Quando em modo treinamento, é possível aplicar o algoritmo de reconstrução de imagens, capaz de alinhar quadro a quadro cada imagem permitindo reconstruir todo o ambiente adquirido, e não apenas um ponto de vista do mesmo. Por sua vez, em modo teste as imagens não são reconstruídas.

Algoritmos que trabalham com nuvem de pontos são em sua grande maioria computacionalmente custosos, sendo que normalmente a sua complexidade está relacionada a quantidade de pontos presentes na nuvem. Por meio da aplicação da técnica de *Downsampling* foi possível reduzir drasticamente o número de pontos da nuvem capturada em cerca de 90%, sem perda

considerável de informação.

Seguindo o *pipeline* de aprendizado *3D* proposto, a próxima etapa consiste na segmentação. Após a obtenção da nuvem de pontos, não existe nenhum conceito de limite. O objetivo da segmentação consiste em "quebrar" a nuvem de pontos em segmentos que representem objetos ou subobjetos.

Segmentação de imagens é um passo essencial no processamento e reconhecimento de imagem. Uma falha na segmentação pode causar a união de objetos distintos ou adicionar ruído. Uma vez que o resultado da segmentação é utilizado como entrada para outros algoritmos, é importante que este seja rápido, flexível e particularmente confiável. Por meio de uma heurística que considera localidade espacial e geometria, a segmentação foi realizada com êxito. Ao fornecer uma nuvem de pontos ao algoritmo de segmentação, como resultado são gerados *n* arquivos, onde cada um dos 1...*n* arquivos representam um *cluster* obtido da nuvem de pontos. Os experimentos e os resultados gerados mostram que o algoritmo se comportou melhor que o algoritmo *"Euclidean Cluster Extraction"* presente na PCL.

A próxima etapa consiste em representar um objeto utilizando o modelo proposto e buscar por características que consigam identificar a nuvem de pontos e extraí-las. Então, com os objetos já representados e com características que permitam o identificar, estudar algoritmos de aprendizado de máquina que permitam realizar o treinamento e teste. Este trabalho provê um *framework* que poderá ser utilizado como base para reconhecimento de objetos, de maneira que o modelo de representação proposto proporcione facilidades na implementação de novas características para trabalhos futuros.

5.1 Contribuições

A contribuição primária deste trabalho consiste no estudo de aquisição, representação e reconhecimento de padrões sobre estruturas tridimensionais. A ideia inicial era adquirir os dados por meio de câmeras *stereo*. Entretando os dados obtidos por câmeras *stereo* não eram suficientemente bons, e então tentou-se obter resultados mais satisfatórios por meio da técnica de algoritmo genético com o objetivo de otimizar os parâmetros da chamada da função do *OpenCV*. Mesmo com a melhora nos resultados, ainda foram considerados fracos para aplicações reias. Foi pensado então na aquisição dos dados com o *Kinect*, que foi realizado com sucesso.

Após o passo de aquisição dos dados, foi realizado também um estudo sobre reconhecimento de padrões em representações tridimensionais. Dois estudos em paralelo foram realizados. O primeiro foi o reconhecimento de padrões dos dados tridimensionais obtidos do *wiimote*

com o objetivo de reconhecer movimento. Conforme apresentado no trabalho, os resultados obtidos foram considerados um sucesso. A grande maioria dos movimentos foi reconhecido com uma precisão maior que 95% e todos realizados em tempo real. Assim, este estudo mostrou que com a utilização da técnica adequada, mesmo trabalhando-se com uma grande quantidade de dados é possível alcançar altas taxas de acerto juntamente com tempo de execução satisfatório.

O segundo foi realizar o reconhecimento de objetos com os dados adquiridos do *Kinect*. Foi proposto um *pipeline* de aprendizado *3D*. Nesse *pipeline* foi proposto ainda as técnicas de *downsampling*, segmentação e o conceito de representação de um objeto real como um objeto computacional por meio de geração automática de código durante a fase de anotação.

A conceito de representação consiste no fato de propor uma representação mais "alto nível" para a representação do objeto adquirido. Este não seria representado apenas como um conjunto de pontos *3D*, mas sim um objeto computacional. A ideia de abstrair a representação do objeto permite, por exemplo, maior naturalidade na programação de novas rotinas para os objetos, descritores entre outras facilidades.

5.2 Trabalhos Futuros

Esse trabalho apresentou um fomento de assuntos que permitem contribuições futuras sobre diversas frentes. Cada item do *pipeline de aprendizado* pode ser estudado e melhorado. Como principais, pode-se melhorar a técnica de *downsampling* permitindo que a nuvem de pontos seja reduzida de forma dinâmica, ou seja, o algoritmo consiga ajustar a redução dos pontos durante a execução por meio da identificação de uma região com muita informação ou não, evitando assim grande reduções em regiões onde se concentra uma grande quantidade de informação.

Neste trabalho, utiliza-se a técnica de segmentação com conceitos de localidade espacial, geometria e coloração. Na literatura existem outras abordagens para segmentação de imagens, como por exemplo, a detecção e utilização de bordas para caracterizar objetos. É interessante como trabalho futuro realizar um estudo das abordagens de segmentação existentes.

Foi proposto também um *framework* para reconhecimento de objetos. Porém, os resultados obtidos ao longo da pesquisa não foram conclusivos quanto ao reconhecimento de objeto. Como trabalho futuro é interessante um estudo mais amplo sobre descritores de nuvens de pontos que permitam caracterizar fortemente uma nuvem de ponto. Portanto, é de interesse que se estude e utilizando-se do *framework* avalie a precisão do reconhecimento de objeto.

Além disso pode-se estudar e melhorar a forma de representar objetos reais como objetos

computacionais, podendo propor outros modelos que melhor se adequem ao problema.

Referências Bibliográficas

BAHLMANN, C.; BURKHARDT, H. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 26, n. 3, p. 299–310, 2004.

BARIYA, P.; NISHINO, K. Scale-hierarchical 3d object recognition in cluttered scenes. In: IEEE. *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. [S.l.], 2010. p. 1657–1664.

BEASLEY, D.; MARTIN, R.; BULL, D. An overview of genetic algorithms: Part 1. fundamentals. *University computing*, Citeseer, v. 15, p. 58–58, 1993.

BEAUCHEMIN, S.; BARRON, J. The computation of optical flow. *ACM Computing Surveys (CSUR)*, ACM, v. 27, n. 3, p. 433–466, 1995.

BILLON, R.; NÉDÉLEC, A.; TISSEAU, J. Gesture recognition in flow based on PCA analysis using multiagent system. In: ACM. *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*. [S.l.], 2008. p. 139–146.

BOGUE, E. T.; MATSUBARA, E. T.; BESSA, A. C. Uma abordagem em reconhecimento de movimentos utilizando trknn e dynamic time warping. In: *BRACIS - Brazilian Conference on Intelligent Systems*. [S.l.: s.n.], 2012.

BRACHMAN, R.; LEVESQUE, H. *Knowledge Representation and Reasoning (The Morgan Kaufmann Series in Artificial Intelligence)*. 1. ed. [S.l.]: Morgan Kaufmann, 2004. ISBN 9781558609327.

BRADSKI, G.; KAEHLER, A. *Learning OpenCV: Computer vision with the OpenCV library*. [S.l.]: O'Reilly Media, 2008.

BURRUS, N. *Kinect Calibration*. May 2012. Http://nicolas.burrus.name/index.php/Research/KinectCalibration.

BURRUS, N. *Open Kinect*. July 2012. Http://openkinect.org/wiki/Main_Page.

CAIANI, E. et al. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. In: IEEE. *Computers in Cardiology 1998*. [S.l.], 1998. p. 73–76.

CHEN, H.; BHANU, B. Efficient recognition of highly similar 3d objects in range images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 31, n. 1, p. 172–179, 2009.

CHU, P.; BEASLEY, J. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, Springer Netherlands, v. 4, p. 63–86, 1998. ISSN 1381-1231. 10.1023/A:1009642405419. Disponível em: <http://dx.doi.org/10.1023/A:1009642405419>.

COLLET, A. et al. Object recognition and full pose registration from a single image for robotic manipulation. In: IEEE. *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. [S.l.], 2009. p. 48–55.

COLMERAUER, A.; ROUSSEL, P. The birth of prolog. In: ACM. *History of programming languages—II*. [S.l.], 1996. p. 331–367.

DAVIS, R.; SHROBE, H.; SZOLOVITS, P. What is a knowledge representation? *AI magazine*, v. 14, n. 1, p. 17, 1993.

DOTNETNUKE CORPORATION. *Open Natural Interface*. July 2012. Http://75.98.78.94/.

EKVALL, S.; KRAGIC, D.; HOFFMANN, F. Object recognition and pose estimation using color cooccurrence histograms and geometric modeling. *Image and Vision Computing*, Elsevier, v. 23, n. 11, p. 943–955, 2005.

FAYED, H.; ATIYA, A. A Novel Template Reduction Approach for the *K*-Nearest Neighbor Method. *Neural Networks, IEEE Transactions on*, IEEE, v. 20, n. 5, p. 890–896, 2009. ISSN 1045-9227.

FISCHLER, M.; BOLLES, R. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, ACM, v. 24, n. 6, p. 381–395, 1981.

FROME, A. et al. Recognizing objects in range data using regional point descriptors. *Computer Vision-ECCV 2004*, Springer, p. 224–237, 2004.

GALLO, L.; PLACITELLI, A.; CIAMPI, M. Controller-free exploration of medical image data: Experiencing the kinect. In: IEEE. *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on*. [S.l.], 2011. p. 1–6.

GONG, M.; YANG, Y. Genetic-based stereo algorithm and disparity map evaluation. *International Journal of Computer Vision*, Springer, v. 47, n. 1, p. 63–77, 2002.

GRANGER, E. M. *Is CIE L*a*b* Good Enough for Desktop Publishing?* [S.l.], 1994.

HAN, K. et al. Stereo matching using genetic algorithm with adaptive chromosomes. *Pattern Recognition*, Elsevier, v. 34, n. 9, p. 1729–1740, 2001.

HEMPEÎ, C. Aspects of scientific explanation. 1965.

HENGSTLER, S. *Stereo Vision in Smart Camera Networks*. November 2008.

HENRY, P. et al. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In: *the 12th International Symposium on Experimental Robotics (ISER)*. [S.l.: s.n.], 2010. v. 20, p. 22–25.

HIRSCHMULLER, H.; SCHARSTEIN, D. Evaluation of cost functions for stereo matching. In: IEEE. *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. [S.l.], 2007. p. 1–8.

HONG, P.; HUANG, T.; TURK, M. Gesture modeling and recognition using finite state machines. *fg*, Published by the IEEE Computer Society, p. 410, 2000.

HUANG, Y.; ZHUANG, X. Motion-partitioned adaptive block matching for video compression. In: IEEE. *Image Processing, 1995. Proceedings., International Conference on*. [S.l.], 1995. v. 1, p. 554–557.

JIANG, X.; BUNKE, H. Fast segmentation of range images into planar regions by scan line grouping. *Machine vision and applications*, Springer, v. 7, n. 2, p. 115–122, 1994.

KEOGH, E.; RATANAMAHATANA, C. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, Springer, v. 7, n. 3, p. 358–386, 2005. ISSN 0219-1377.

KEOGH, E.; RATANAMAHATANA, C. Exact indexing of dynamic time warping. *Knowledge and information systems*, Springer, v. 7, n. 3, p. 358–386, 2005.

KIM, J.; MASTNIK, S.; ANDRÉ, E. Emg-based hand gesture recognition for realtime biosignal interfacing. ACM, New York, NY, USA, p. 30–39, 2008. Disponível em: <http://doi.acm.org/10.1145/1378773.1378778>.

KIRBY, M.; SIROVICH, L. Application of the karhunen-loeve procedure for the characterization of human faces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 12, n. 1, p. 103–108, 1990.

KJAER, J. *A Qualitative Analysis of Two Automated Registration Algorithms In a Real World Scenario Using Point Clouds from the Kinect*. June 2011. Bachelor Thesis.

KLEMENT, K. C. *Propositional Logic*. July 2005. Http://www.iep.utm.edu/prop-log/.

KLETTE, R.; SCHLÜNS, K.; KOSCHAN, A. *Computer vision: three-dimensional data from images*. [S.l.]: Springer Singapore, 1998.

KOVACS-VAJNA, Z. A fingerprint verification system based on triangular matching and dynamic time warping. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 22, n. 11, p. 1266–1276, 2000.

KOWALSKI, R. The early years of logic programming. *Communications of the ACM*, ACM, v. 31, n. 1, p. 38–43, 1988.

LAI, K. et al. A large-scale hierarchical multi-view rgb-d object dataset. In: *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*. [S.l.: s.n.], 2011.

LAI, K. et al. Sparse distance learning for object recognition combining rgb and depth information. In: IEEE. *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. [S.l.], 2011. p. 4007–4013.

LARRAñAGA, P. et al. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, v. 13, p. 129–170, 1999.

LOWE, D. Object recognition from local scale-invariant features. In: IEEE. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. [S.l.], 1999. v. 2, p. 1150–1157.

LOWE, D. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, Springer, v. 60, n. 2, p. 91–110, 2004.

MANGAN, A.; WHITAKER, R. Partitioning 3d surface meshes using watershed segmentation. *Visualization and Computer Graphics, IEEE Transactions on*, IEEE, v. 5, n. 4, p. 308–321, 1999.

MATTHIES, L.; KANADE, T.; SZELISKI, R. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, Springer, v. 3, n. 3, p. 209–238, 1989.

MCCARTHY, J. et al. *LISP 1.5 Programmer's Manual*. [S.l.], 1962.

MCCARTHY, J. et al. *LISP I Programmers Manual*. [S.l.], March 1960.

MERRITT, D. *Building expert systems in Prolog*. [S.l.]: Springer-Verlag, 1989.

MICROSOFT CORPORATION. *Kinect for Windows*. July 2012. Http://www.microsoft.com/en-us/kinectforwindows/.

MICROSOFT CORPORATION. *Kinect for Windows Sensor Components and Specifications*. July 2012. Http://msdn.microsoft.com/en-us/library/jj131033.aspx.

MITCHELL, T. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 1997.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997.

MITRA, S.; ACHARYA, T. Gesture recognition: A survey. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, IEEE, v. 37, n. 3, p. 311–324, 2007. ISSN 1094-6977.

MITTRAPIYANURUK, P.; DESOUZA, G.; KAK, A. Calculating the 3d-pose of rigid-objects using active appearance models. In: IEEE. *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. [S.l.], 2004. v. 5, p. 5147–5152.

MOKHTARIAN, F.; KHALILI, N.; YUEN, P. Multi-scale free-form 3d object recognition using 3d models. *Image and Vision Computing*, Elsevier, v. 19, n. 5, p. 271–281, 2001.

MÜHLMANN, K. et al. Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision*, Springer, v. 47, n. 1, p. 79–88, 2002.

MURAKAMI, K.; TAGUCHI, H. Gesture recognition using recurrent neural networks. In: ACM. *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*. [S.l.], 1991. p. 237–242. ISBN 0897913833.

MYERS, C.; RABINER, L. A level building dynamic time warping algorithm for connected word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, IEEE, v. 29, n. 2, p. 284–297, 1981.

NAIN, D. et al. Shape-driven 3d segmentation using spherical wavelets. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2006*, Springer, p. 66–74, 2006.

ORACLE. *Java*. July 2012. Www.java.com.

PAL, N.; PAL, S. A review on image segmentation techniques. *Pattern recognition*, Elsevier, v. 26, n. 9, p. 1277–1294, 1993.

POINTCLOUD.ORG. *Point Cloud Library*. July 2012. Http://pointclouds.org/.

POLI, G. et al. Voice command recognition with dynamic time warping (dtw) using graphics processing units (gpu) with compute unified device architecture (cuda). IEEE Computer Society, 2007. ISSN 1550-6533.

PORTER, B. *Handbook of knowledge representation*. [S.l.]: Elsevier Science, 2008.

PYTHON SOFTWARE FOUNDATION. *Python FAQS*. April 2012. Http://docs.python.org/faq/general.html#what-is-python-good-for.

PYTHON SOFTWARE FOUNDATION. *Python Programming Language*. July 2012. Http://www.python.org.

RABBANI, T.; HEUVEL, F. van D.; VOSSELMANN, G. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, v. 36, n. 5, p. 248–253, 2006.

RABINER, L.; JUANG, B. *Fundamentals of speech recognition*. [S.l.]: Prentice hall, 1993.

RATH, T.; MANMATHA, R. Word image matching using dynamic time warping. In: IEEE. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. [S.l.], 2003. v. 2, p. II–521.

RIDEL, D. A. et al. Calibraçao de visao estereoscópica utilizando algoritmo genético e dtw. In: *BRACIS - Brazilian Conference on Intelligent Systems*. [S.l.: s.n.], 2012.

RIEMERSM, T. *Colour Metric*. June 2012. Http://www.compuphase.com/cmetric.htm.

ROBINSON, J. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, ACM, v. 12, n. 1, p. 23–41, 1965.

RUSSELL, B. et al. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, Springer, v. 77, n. 1, p. 157–173, 2008.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach (3rd Edition)*. 3. ed. [S.l.]: Prentice Hall, 2009. ISBN 9780136042594.

RUSU, R.; BLODOW, N.; BEETZ, M. Fast point feature histograms (fpfh) for 3d registration. In: IEEE. *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. [S.l.], 2009. p. 3212–3217.

RUSU, R. et al. Fast 3d recognition and pose using the viewpoint feature histogram. In: IEEE. *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. [S.l.], 2010. p. 2155–2162.

RUSU, R. et al. Learning informative point classes for the acquisition of object model maps. In: IEEE. *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*. [S.l.], 2008. p. 643–650.

RUSU, R. et al. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, Elsevier, v. 56, n. 11, p. 927–941, 2008.

RUSU, R. B. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. Tese (Doutorado) — Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

RZIZA, M. et al. Estimation and segmentation of a dense disparity map for 3d reconstruction. In: IEEE. *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. [S.l.], 2000. v. 6, p. 2219–2222.

SAPPA, A.; DEVY, M. Fast range image segmentation by an edge detection strategy. In: IEEE. *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. [S.l.], 2001. p. 292–299.

SCHARSTEIN, D.; PAL, C. Learning conditional random fields for stereo. In: IEEE. *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. [S.l.], 2007. p. 1–8.

SCHARSTEIN, D.; SZELISKI, R. High-accuracy stereo depth maps using structured light. In: IEEE. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. [S.l.], 2003. v. 1, p. I–195.

SCHLOMER, T. et al. Gesture recognition with a Wii controller. In: ACM. *Proceedings of the 2nd international conference on Tangible and embedded interaction*. [S.l.], 2008. p. 11–14.

SCHMID, C.; MOHR, R. Local grayvalue invariants for image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 19, n. 5, p. 530–535, 1997.

SNAGGLEPANTS. *Python Color Math Mode*. July 2012. Http://code.google.com/p/python-colormath/.

STICKEL, M. A prolog technology theorem prover: Implementation by an extended prolog compiler. *Journal of Automated reasoning*, Springer, v. 4, n. 4, p. 353–380, 1988.

STOWERS, J.; HAYES, M.; BAINBRIDGE-SMITH, A. Quadrotor helicopter flight control using hough transform and depth map from a microsoft kinect sensor,". In: *Proceedings of the IAPR Conference on Machine Vision Applications, Nara, Japan*. [S.l.: s.n.], 2011. p. 352–356.

TAYLOR, G.; KLEEMAN, L. Fusion of multimodal visual cues for model-based object tracking. In: *Australasian conference on robotics and automation (ACRA2003)*. [S.l.: s.n.], 2003.

TURK, M.; PENTLAND, A. Eigenfaces for recognition. *Journal of cognitive neuroscience*, MIT Press, v. 3, n. 1, p. 71–86, 1991.

VONDRICK, C.; RAMANAN, D.; PATTERSON, D. Efficiently scaling up video annotation with crowdsourced marketplaces. *Computer Vision–ECCV 2010*, Springer, p. 610–623, 2010.

VOSSELMAN, G. et al. Recognising structure in laser scanner point clouds. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, v. 46, n. 8, p. 33–38, 2004.

WÄHLBY, C. et al. Combining intensity, edge and shape information for 2d and 3d segmentation of cell nuclei in tissue sections. *Journal of Microscopy*, Wiley Online Library, v. 215, n. 1, p. 67–76, 2004.

WHITLEY, D. A genetic algorithm tutorial. *Statistics and computing*, Springer, v. 4, n. 2, p. 65–85, 1994.

Wikimedia Foundation. *kd-tree*. July 2012. Http://en.wikipedia.org/wiki/K-d_tree.

XIA, L.; CHEN, C.-C.; AGGARWAL, J. Human detection using depth information by kinect. In: IEEE. *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. [S.l.], 2011. p. 15–22.

XIANG, R.; WANG, R. Range image segmentation based on split-merge clustering. In: IEEE. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. [S.l.], 2004. v. 3, p. 614–617.

YU, K.; MASON, J.; OGLESBY, J. Speaker recognition using hidden markov models, dynamic time warping and vector quantisation. In: IET. *Vision, Image and Signal Processing, IEE Proceedings-*. [S.l.], 1995. v. 142, n. 5, p. 313–318.

ZHANG, H. et al. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In: IEEE. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. [S.l.], 2006. v. 2, p. 2126–2136.

ZHONG, Y. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In: IEEE. *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. [S.l.], 2009. p. 689–696.

ZICKLER, S.; VELOSO, M. Detection and localization of multiple objects. In: IEEE. *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. [S.l.], 2006. p. 20–25.