UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Günter Nogueira Loch

IMPLEMENTAÇÃO DE UM FRAMEWORK DE SIMULAÇÃO PARA TESTES DE TÉCNICAS DE VISÃO COMPUTACIONAL APLICADAS À ROBÓTICA MÓVEL

Florianópolis

2013

Günter Nogueira Loch

IMPLEMENTAÇÃO DE UM FRAMEWORK DE SIMULAÇÃO PARA TESTES DE TÉCNICAS DE VISÃO COMPUTACIONAL APLICADAS À ROBÓTICA MÓVEL

Dissertação submetida à Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Mestre em Engenharia de Automação e Sistemas.

Florianópolis

2013

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Loch, Günter Nogueira
Implementação de um framework de simulação para testes de técnicas de visão computacional aplicadas à robótica móvel / Günter Nogueira Loch ; orientador, Marcelo Ricardo Stemmer - Florianópolis, SC, 2013.
110 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Inclui referências

1. Engenharia de Automação e Sistemas. 2. Reconhecimento de objetos. 3. SIFT. 4. robótica móvel. 5. framework de simulação. I. Stemmer, Marcelo Ricardo. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Titulo.

Günter Nogueira Loch

IMPLEMENTAÇÃO DE UM FRAMEWORK DE SIMULAÇÃO PARA TESTES DE TÉCNICAS DE VISÃO COMPUTACIONAL APLICADAS À ROBÓTICA MÓVEL

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Engenharia de Automação e Sistemas", e aprovada em sua forma final pela Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 18 de setembro de 2013.

Prof. Dr. Jomi Fred Hübner
Coordenador do Curso

Banca Examinadora:

Prof. Dr. Marcelo Ricardo Stemmer
Presidente

Dr^{a}. Analucia Vieira Fantin

Prof. Dr. Edson Roberto de Pieri

Prof. Dr. Ubirajara Franco Moreno

AGRADECIMENTOS

Agradeço primeiramente a minha mãe, Ruth Emilia Nogueira, pelo inestimável suporte, sem o qual não seria possível concluir este trabalho de Mestrado. Agradeço também os colegas do PGEAS e principalmente aos amigos Andreu Carminati, Renan Starke, José Gilmar de Carvalho, Luciano Barreto e Rodrigo Lange, cujo o conhecimento e amizade foram essenciais para o desenvolvimento desta dissertação. Por último agradeço ao professor Marcelo Stemmer e aos colegas do S2i pelo apoio financeiro necessário para a execução da pesquisa apresentada neste documento.

RESUMO

Neste trabalho foram desenvolvidos um *framework* para construção de simuladores de ambientes tridimensionais e meios para construir bases de dados para um sistema de reconhecimento de objetos para ser aplicado à robótica móvel.

Inicialmente foi feita uma análise de dois ambientes de desenvolvimento de robôs (ADRs) capazes de fornecer maneiras para se construir simulações. Esta avaliação foi feita através da adaptação da metodologia de Kramer e Scheutz (2007) de acordo com as necessidades relacionadas ao desenvolvimento de aplicações de visão computacional. Como resultado, percebeu-se que ambos os ADRs deixavam a desejar quanto à qualidade gráfica disponibilizada por padrão. Desta forma, foi necessário buscar soluções para este problema, o que resultou no desenvolvimento do *framework* para implementação de simuladores tridimensionais de alta fidelidade, usando o motor de renderização OGRE (*Object-Oriented Graphics Rendering Engine*).

Posteriormente foi feito o estudo relacionado ao descritor de características locais SIFT (*Scale-Invariant Feature Transform*). Este estudo abordou o próprio SIFT e o método de reconhecimento de objetos baseado nele, proposto por Lowe (2004), o qual foi avaliado do ponto de vista da aplicação na robótica móvel. A partir desta avaliação, verificou-se a necessidade de aplicar o algoritmo de agrupamento tridimensional de poses de Lowe (2001) no processo de geração de bases de dados para o sistema de reconhecimento de objetos, como forma de aumentar a robustez deste quanto à variação de pose. Este algoritmo representa objetos por meio de modelos de pose, os quais são ligados se existem correspondências entre as poses. A análise deste algoritmo revelou as variáveis que afetam o processo de geração das bases de dados, o que permitiu a proposição de uma nova estratégia, baseada em recálculo, para a criação das ligações que o algoritmo de agrupamento faz ao correlacionar duas poses.

Através da implementação do sistema de construção de bases de dados, baseado no algoritmo de agrupamento de poses, sobre o *framework* de simulação, foi possível avaliar sinteticamente o efeito da modificação das variáveis nas bases de dados criadas, observando que a estratégia proposta consegue distribuir melhor as ligações entre os modelos de pose de um objeto, em relação ao método original.

Palavras-chave: Reconhecimento de objetos, SIFT, robótica móvel, framework de simulação

ABSTRACT

On this work, it was developed a framework to build three-dimensional environment simulators, and implemented methods to build databases for a object recognition system, which should be applied in mobile robots.
Initially, two robotic development environments (RDEs) were evaluated, which were able to provide a way to build simulations. This assessment was done through modifications in the methodology proposed by Kramer e Scheutz (2007), in order to fit it, to the needs related to the development of computer vision applications. As result, it was realized that both RDEs, in their default installation, did not attain the desired graphical standard. The necessity of solutions to this problem triggered a search for them, which result in the development of the framework to implement high-fidelity tridimensional simulators, using the OGRE (Object-Oriented Graphics Rendering Engine).
Then, a review of the SIFT (Scale-Invariant Feature Transform) local feature descriptor was done. This review assessed the descriptor itself and the object recognition method proposed by Lowe (2004), considering the standpoint of a mobile robot application. From this assessment, it was noticed the need to apply the 3D view clustering algorithm proposed by Lowe (2001) to the database generation process, in order to improve the robustness of this relative to viewpoint changes.
This algorithm represents objects through clustering model views, which are linked if there are matches between them. This algorithm represents objects through clustering model views, which are linked if there are matches between them. Though the analysis of the algorithm, the variables that could affect the database generation process were noticed, and a new way to build the links that the algorithm creates when it matches two images was proposed. This new approach is based on recalculations of the links.
Ultimately, the effects of varying the variables of the clustering algorithm were assessed synthetically. This evaluation was done through tests using the implementations of database generation system based on the clustering algorithm. It was noticed that the new way to build links is able to disperse them in a better way, when compared to original method.

Keywords: Object Recognition, SIFT, mobile robots, simulation framework

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Aumentar as habilidades cognitivas de robôs é um interesse constante na comunidade científica ligada ao campo da robótica. Deseja-se com isso, que sistemas robóticos possam adquirir maior nível de flexibilidade, característica esta, típica de seres humanos. Um exemplo de uma ação simples para uma pessoa, mas extremamente complexa para um robô, seria pedir para que cumprisse a seguinte ordem:

"*Vá a segunda **mesa**, ao lado esquerdo do **armário** de ferramentas, e pegue o **martelo***".

Se a pessoa do exemplo sabe a localização do referido armário, a tarefa se torna tão simples, que esta nem percebe o raciocínio envolvido. De forma resumida, isto aconteceria da seguinte maneira: inicialmente a pessoa se deslocará para onde o armário está localizado. Ao perceber este, a pessoa avança o raciocínio à procura das duas mesas, ancorando sua decisão na percepção obtida. A busca pelo martelo sobre a mesa segue a mesma lógica: ela avança a busca do martelo ancorando-se na percepção da localização das mesas. Assim, é fácil notar, neste exemplo, a importância do reconhecimento de objetos para o cumprimento das mais simples tarefas.

Para reconhecer os objetos, usamos principalmente o sentido da visão. Este também tem função preponderante ao desempenhar os deslocamentos necessários para alcançar cada um dos objetos do exemplo anterior. Como não entendemos completamente o funcionamento do sistema de visão humano, é muito difícil replicá-lo. Além disso, nem todas características deste sistema são interessantes, como a suscetibilidade a ilusões de ótica.

Mesmo assim, as capacidades que a visão oferece aos seres humanos fazem com que haja interesse na pesquisa da aplicação da visão computacional em vários problemas da robótica. Desta compreensão nasceu esta pesquisa de mestrado, a qual tem como foco a busca e aplicação de algoritmos de reconhecimento de objeto para solução de problemas de movimentação robótica. Devido ao tamanho deste desafio, foi necessário definir e delimitar os aspectos abordados nesta dissertação

O primeiro passo foi desenvolver um simulador específico para o caso, pois tal situação poderia propiciar um controle maior as variáveis dos testes, permitindo observações mais precisas, seja na variação de parâmetros ou na comparação de técnicas. Outros pesquisadores já observaram estas dificuldades na execução de testes, levando-os a desenvolver seus simuladores e ambientes de desenvolvimento para robótica (ADR). Conforme mostram Kramer e Scheutz (2007), tem havido grande desenvolvimento deste tipo de ferramentas. Hoje a indústria da computação gráfica já consegue criar mo-

delos renderizados fidedignos à aparência real. Tal fato torna possível pensar em simular ambientes de testes com um significativo grau de fidelidade na aparência. Todos estes fatores levaram à decisão de pesquisar quanto à execução de testes de algoritmos de visão computacional através deste tipo de ferramenta.

Já na parte relacionada a visão computacional, verificou-se que atualmente existe um interesse em algoritmos baseados em descritores de características locais. Foi encontrado farto material relacionado a este tipo de técnica, com aplicações e comparativos nas áreas de reconhecimento de objetos e padrões, e robótica móvel. Em especial, verificou-se o elevado número de aplicações do descritor SIFT (*Scale-Invariant Feature Transform*), proposto inicialmente por Lowe (1999). Morel (2009) afirma que o SIFT tem se mostrado superior a outros descritores como *distribution-based shape context*, *geometric histogram*, *derivative-based complex filters*, e *moment invariants*. Considerando isto, as aplicações e os comparativos encontrados, decidiu-se pesquisar sobre a utilização deste algoritmo no reconhecimento de objetos e navegação de robôs.

Porém, a escolha deste tema também apresentou um dilema: desenvolver o sistema de reconhecimento de objetos, ou as bases de dados para permitir o primeiro. Para que o sistema possa exercer sua função, ele precisa ter o conjunto de informações dos objetos a serem reconhecidos. Já para que se possa avaliar as bases de dados em caráter definitivo, precisa-se do sistema de reconhecimento.

Mas, baseando-se em Ramisa et al. (2009) que se definiu pela escolha da geração de bases de dados como segundo objetivo deste trabalho. Este autor sugere como trabalho futuro o uso do método para agrupamento tridimensional de poses (*3D view clustering*) de Lowe (2001), para aumentar a robustez do sistema de reconhecimento. Considerando tais prerrogativas, decidiu-se investigar este algoritmo na geração de bases de dados para o método de reconhecimento de objetos baseado no SIFT, considerando o agrupamento de diferentes vistas do mesmo objeto.

Levando em consideração o descrito acima, foram definidos os seguintes objetivos principais e específicos para este trabalho:

1. Desenvolver um ambiente de testes onde se possa simular tridimensionalmente, um robô móvel dotado de câmeras.

    1.1 Analisar, por meio da metodologia proposta por Kramer e Scheutz (2007), ambientes de desenvolvimento de robôs (ADRs) que fornecem meios de simulação.

    1.2 Implementar o ambiente de testes, por meio dos conhecimentos obtidos nesta fase de análise.

1.3 Avaliar o ambiente desenvolvido através da análise da emulação de um sistema de câmeras estéreo acoplado a um robô, por meio de teste de calibração.

2. Iniciar a implementação de um sistema de reconhecimento de objetos baseado no SIFT, elaborando a parte referente a geração das base de dados.

   2.1 Implementar o sistema para geração de base de dados e o mecanismo necessário para extração de imagens dos objetos no ambiente de testes.

   2.2 Identificar as variáveis do algoritmo de agrupamento tridimensional de poses que podem afetar a criação de bases de dados.

   2.3 Avaliar sinteticamente como alterações nestas variáveis afetam o resultado da criação das bases de dados.

A fim de descrever o trabalho realizado, a escrita deste documento foi divida em uma estrutura de capítulos. Primeiramente, o capítulo 1, refere-se a esta introdução. O capítulo 2 mostra uma revisão bibliográfica e conceitual. Foi dado ênfase na compreensão do problema de movimentação robótica, definindo inicialmente os conceitos relacionados a área. Também são apresentadas neste capítulo, as classificações para sistemas de movimentação de robôs, e publicações relacionadas a algumas das classes citadas. Na sequência foi feita uma revisão de conceitos relacionados a visão computacional para posteriormente mostrar artigos descrevendo algumas soluções da área de visão computacional para os problemas de movimentação de robôs. Destaca-se o trabalho de Kim e Nevatia (1999), o qual demonstrou um sistema de navegação simbólica baseado em visão computacional. Por último, neste capítulo, verificou-se a aplicação de algoritmos de descrição local de características, mostrando artigos comparativos e de aplicações.

No capítulo 3 abordou-se a construção do ambiente de simulação. Primeiramente foi apresentada a metodologia usada para analisar os ADRs avaliados. Com os estudos destas ferramentas, percebeu-se a necessidade de implementar um *framework* próprio, para o desenvolvimento de simuladores para teste de aplicações visuais. A implementação foi descrita em duas seções. A primeira é relativa ao desenvolvimento do *framework* e do simulador, enquanto a segunda refere-se aos aspectos de computação gráfica envolvidos na criação de um simulador tridimensional de alta fidelidade.

O capítulo 4 trata do desenvolvimento das bases de dados para o futuro sistema de reconhecimento de objetos. Inicialmente foi feita a apresentação do descritor SIFT. Depois, a descrição do sistema de reconhecimento de objetos baseado no SIFT, proposto por Lowe (2004). Na sequência foi

apresentado o método de agrupamento utilizado, discutindo sua aplicação e as variáveis que afetam a construção das bases de dados. Por último, foram abordadas as implementações necessárias.

Por fim, o capítulo 5 apresenta os resultados obtidos tanto com relação ao desenvolvimento do ambiente de teste, quanto dos resultados relacionados a parte de visão.

As considerações relativas a trabalhos futuros e discussões finais estão confinadas no capítulo 6.

2 REVISÃO BIBLIOGRÁFICA E CONCEITUAL

Neste capítulo são abordados inicialmente alguns conceitos importantes da robótica móvel, definindo-os. Na sequência, são apresentadas duas classificações para sistemas de movimentação de robôs. Em seguida, é apresentado uma revisão de conceitos de visão computacional para posteriormente apresentar o estudo feito quanto a sistemas de movimentação baseados em visão computacional. Primeiramente são apresentadas três aplicações de visão computacional na movimentação de robôs, dando ênfase ao trabalho utilizando navegação simbólica.

Além disso, são apresentados trabalhos relacionados à área de descrição local de pontos de interesse aplicadas à robótica e ao reconhecimento de objetos. Há duas referências quanto à avaliação de descritores: a primeira no campo de mapeamento e localização simultânea e a segunda na aplicação de reconhecimento de objetos à robótica móvel. Por fim, duas aplicações à robótica móvel do descritor local mais promissor são mostradas.

2.1 CONCEITOS RELACIONADOS À ROBÓTICA MÓVEL

Robôs móveis e veículos autônomos representam um campo de pesquisa que aborda um amplo espectro de aplicações. As diversas aplicações possuem diferentes tipos de requisitos que por sua vez podem ser solucionados de várias maneiras. Uma das causas da grande variedade de soluções está ligada a diversas formações e/ou objetivos dos pesquisadores da área. Como consequência, certos conceitos têm sentidos ambíguos e/ou diferenciados de acordo com o problema ou área de pesquisa da robótica móvel.

Por exemplo, pesquisadores da área de sensores tendem a tratar o conceito de navegação de modo mais amplo, com o sentido de movimentação usado pelos pesquisadores da área de planejamento de movimento. Devido a necessidade de ter maior rigor na caracterização destes conceitos, será usada neste trabalho a definição da área de planejamento de movimento. Choset et al. (2005) caracterizam um sistema de movimentação de acordo com : a **tarefa** que o robô deve atender, as **propriedades** do mesmo e do ambiente, e o **algoritmo** utilizado. Essa caracterização está resumida na Tabela 1.

a) Tarefa

A característica mais importante de um sistema de movimentação é a tarefa a qual esse se propõe a solucionar. Do ponto de vista de locomoção, uma tarefa pode ser classificada como: *navegação*, *localização*, *cobertura* ou *ma-*

Tabela 1 – Conceitos relacionados a robótica móvel Choset et al. (2005).

Tarefa	**Propriedades**	**Algoritmo**
Navegação	Graus de Liberdade, Espaço	Movimentação (não)ótima
Mapeamento	de Configuração	Complexidade
Cobertura	Cinemática/Dinâmica	Completude
Localização	Restrições de movimento/	Online/Offline
	Omnidirecional	Sensoreamento/
		modelo do Ambiente

peamento. A *navegação* é o problema de encontrar um movimento livre de colisão, que leve um robô de um estado a outro. Normalmente está ligada a robôs móveis, porém também é utilizada na definição de trajetórias para braços robóticos. *Localização* é o problema de determinar a configuração (isto é, o estado instantâneo) do robô. *Cobertura* é o problema de passar por todos os estados do sistema. No caso de um robô móvel, seria a tarefa deste passar por todas as posições possíveis no ambiente. O *mapeamento* é a tarefa de explorar um ambiente construindo uma representação deste.

Apesar de terem diferentes objetivos, estas tarefas tem um certo grau de interdependência. Para um robô móvel cobrir um ambiente é necessário ter algum tipo de representação deste, de forma a garantir que todos os estados (posições) foram alcançados. Por outro lado, para se criar um mapa completo de um ambiente é necessário que se conheça todo o ambiente, ou seja obtenha-se informações sobre todos os estados possíveis. Ao mesmo tempo, para que o mapa represente o ambiente com exatidão, é necessário que a localização de objetos ou referências presentes neste sejam definidas corretamente. Alguns métodos permitem que o mapeamento e a localização sejam feitas simultaneamente, os quais são conhecidos por *SLAM –Simultaneous localization and mapping*.Resumidamente, a movimentação de um robô depende da tarefa a qual este deve cumprir e da quantidade de informação disponível.

b) Propriedades

O modo como será feita a movimentação de um robô é obviamente bastante dependente das características físicas do mesmo e do ambiente no qual se dará a ação. Em conjunto eles definem o número de *graus de liberdade* e *espaço de configuração* do sistema. A partir destas definições, pode-se inferir a capacidade de movimento do robô no ambiente em questão. Aquele que, na ausência de obstáculos, pode se mover em qualquer direção é dito *Omnidi-*

(a) Rôbo omnidirecional (b) Robô diferencial

Figura 1 – Exemplos de rôbos quanto a propriedades

recional (Figura 1a). Um robô diferencial, como o da figura 1b, geralmente é modelado com uma restrição quanto à movimentação lateral, limitando a velocidade transversal do robô a zero. Robôs com essa característica são ditos *não-holonômicos*. Além disso, a modelagem para fins de controle tem influência na caracterização do movimento, podendo ser feita através da *cinemática* ou da *dinâmica* do robô.

c) Algoritmo

Dada a tarefa e o robô que irá desempenhá-la, resta definir o procedimento a ser executado para solucionar o problema. Como qualquer algoritmo, os métodos desenvolvidos para a movimentação de robôs estão sujeitos a critérios de avaliação: *otimalidade*, *complexidade*, *completude*. No contexto da robótica móvel a *otimalidade* do movimento pode ser avaliada através de grandezas como o deslocamento, o tempo e a energia necessários para se movimentar da origem até a posição desejada. A *complexidade* computacional pode ser compreendida como a ordem de crescimento dos recursos necessários (tempo de execução, memória) para solucionar o problema em função do seu "tamanho". O tamanho de um problema de movimentação robótica depende da descrição deste, isto é, como robô, o ambiente e a interação entre estes são modelados.

Uma característica desejável aos algoritmos, é que estes sempre encontrem uma trajetória que solucione o problema de movimentação ou indiquem a inexistência ou falha em um tempo definido. Um método assim é dito *completo*. A completude em sistemas com grande complexidade pode ser intratável computacionalmente. Neste caso atenua-se o requisito de completude através de probabilidades ou modificações na discretização dos movimentos.

De maneira geral há um compromisso entre estes três critérios, de forma que um aumento nos requisitos em algum deles tem impacto negativo com relação aos outros.

Outras duas características importantes estão relacionadas à maneira como a solução será atingida. Em alguns problemas, o algoritmo cria um plano de ações completo antes de iniciar a movimentação capaz de levar o robô da origem ao local desejado. Este plano é baseado apenas no *modelo do ambiente* conhecido anteriormente. Neste caso se tem um algoritmo *off-line*. Por outro lado, algoritmos *online* são aqueles que definem as ações de maneira incremental, isto é, enquanto o robô está se movendo. Neste caso, o algoritmo pode ser baseado em sensores, intercalando o *sensoriamento*, computação e ação. Na prática, a distinção entre métodos *online* e *offline* se dá através do tempo de computação.

2.2 CLASSIFICAÇÕES DE SISTEMAS DE MOVIMENTAÇÃO

As diversas áreas de conhecimento envolvidas na robótica móvel tendem a criar classificações particulares para os sistemas de movimentação, a partir de um ponto de vista específico. Numa abordagem generalista, Dixon e Henlich (1997)[1] apresentaram duas classificações. A primeira é baseada na escala de movimentação. A segunda fundamenta-se na maneira como os algoritmos identificam as mudanças de estado, isto é, mensuram os movimentos.

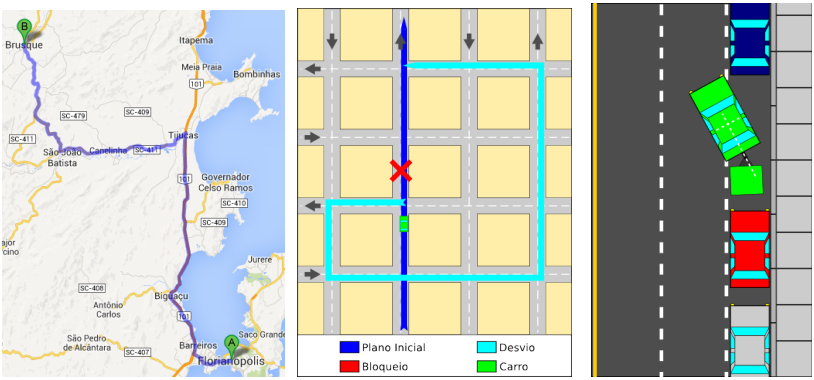
2.2.1 Escala de movimentação

Considerando a escala de movimentação, os sistemas de movimentação podem ser divididos em três tipos:

- Global: Habilidade de determinar sua posição em relação a um mapa ou de modo absoluto com relação à referência, e movimentar-se em direção ao destino.
- Local: Habilidade de determinar a posição em relação a objetos (estáticos ou dinâmicos) no ambiente, e corretamente interagir com estes.
- Personalizada: Estar consciente da posição das diferentes partes que fazem parte de si próprio em relação uma da outra, com o intuito de poder manipular objetos.

[1] As classificações foram modificadas com o intuito de manter a concordância com a definição conceitual usada neste trabalho.

Para exemplificar, busca-se o caso de deslocamentos de um veículo. Os sistemas de navegação global podem ser comparáveis ao problema de encontrar a (melhor) rota em uma viagem intermunicipal (Figura 2a) . A movimentação local pode ser encarada como as ultrapassagens de veículos ou desvios de congestionamentos, por exemplo, (Figura 2b). Neste caso, mesmo que o motorista não conheça o local onde está, ele toma o ponto onde deixou o caminho original e guia-se através do quanto se distanciou deste, até retornar a rota inicial. A movimentação personalizada representa o caso de um motorista de um veículo com reboque, que precise estacionar. Neste caso há a necessidade de levar em conta a articulação da composição para efetuar a tarefa (Figura 2c).



(a) Global (b) Local (c) Personalizada

Figura 2 – Exemplos de movimentação de acordo com a escala de movimento

Para qualquer dos tipos, sempre haverá uma referência para o movimento. Geralmente, determina-se a posição de maneira relativa ou absoluta com relação a uma ou mais referências. Com relação à determinação da posição, o posicionamento absoluto significa encontrar a localização em relação a um ponto estacionário fixo, comum a todo o espaço de navegação. Desta forma para a navegação global é necessário definir uma origem única para o sistema de coordenadas. Para a navegação local, normalmente define-se um ponto fixo no ambiente do robô. No caso da navegação personalizada, geralmente define-se um sistema de coordenadas próprio ao robô.

2.2.2 Algoritmo

Baseando-se nos princípios de funcionamento, os algoritmos de movimentação podem ser classificados de três formas: aqueles baseados em **estimativas**, **marcações** no ambiente e **mapas**. Esta classificação pode ser compreendida em ordem crescente de raciocínio e complexidade, bem como em ordem decrescente de suscetibilidade ao acúmulo de erros. Sistemas reais muitas vezes integram soluções de mais de um tipo, como forma de cobrir as desvantagens inerentes a cada uma das abordagens.

a) Movimentação através de estimativas

A vasta maioria dos robôs móveis baseiam sua estratégia de movimentação em técnicas fundamentadas na estimação de sua trajetória (*Dead-Reckoning*). Como a maioria dos robôs utilizam rodas como meio de locomoção, boa parte deles têm sensores que medem a velocidade e posição angular para estimar sua movimentação. Conhecida como Odometria, esta é a forma mais simples de *Dead-Reckoning*.

Uma alternativa comum aos *encoders*, é utilizar princípios inerciais, mensurando acelerações através de sensores como acelerômetros e giroscópios. Maimone et al. (2007) apresentou o resultado de dois anos de operação dos robôs da missão *Mars Exploration Rovers*. Nestes robôs foram utilizados múltiplos sistemas de odometria, incluindo métodos baseados em visão computacional. As maiores vantagens da movimentação através de estimativas são o custo e a disponibilidade, enquanto a maior desvantagem é o acúmulo de erro.

b) Movimentação através de marcações

Este tipo de estratégia utiliza marcas presentes no ambiente como referência para mensurar deslocamentos ou posição. Pode ser feita através de marcações artificiais ou naturais. A movimentação utilizando marcas artificiais geralmente é mais fácil pois o tamanho, forma e/ou aparência destas são conhecidos *a priori* com exatidão. Além de facilitar a detecção e identificação, essas características permitem uma grande precisão na determinação de posições e distâncias.

Quando se faz uso de aspectos naturais, normalmente usam-se características que possam ser representadas de modo simples e reduzido, como pontos e retas. Neste tipo de sistema, arestas verticais presentes em portas ou na junção de paredes são comumente utilizadas como marcações. Mesmo utilizando características simplificadas, a maior dificuldade está na interpretação correta dos dados dos sensores de forma a identificar corretamente as marcações.

Nebot (1999) apresentou uma série de soluções para este tipo de sistema. Destacam-se o *Bearing-only Laser Scanner* e os Radares de onda milimétrica, que baseiam-se em marcações artificiais na forma de refletores passivos. Madhavan et al. (2002) apresentou um sistema para movimentação em ambientes externos, que utiliza como marcações naturais os pontos de máxima na curvatura do terreno. Neste sistema, a curvatura é obtida através de escaneamento laser, sobre o qual um espaço de escala é construído. A partir do espaço de escala as marcas são definidas de forma invariante à pose do veículo.

c) Movimentação através de mapas

Neste tipo de estratégia, o robô usa seus sensores para construir um mapa local do ambiente, o qual é comparado com um mapa global previamente conhecido. Se uma correspondência é encontrada, então é possível determinar a posição e a orientação do robô com relação ao ambiente. Fica evidente neste caso, que o desempenho da técnica é altamente dependente de como os mapas são construídos e comparados. Estes por sua vez dependem fundamentalmente de como o ambiente é representado.

Há duas representações lógicas mais comuns: Geométrica e Topológica. Mapas geométricos representam objetos de acordo com as relações métricas absolutas destes, por exemplo, através de um mapa de grade. A principal vantagem desta representação está na possibilidade de se obter instantaneamente a relação (distância) entre os diversos objetos do ambiente. Por sua vez, a abordagem topológica registra a relação geométrica entre os objetos, ao invés de manter um referencial arbitrário. Essa característica permite que mapas topológicos sejam construídos e mantidos sem qualquer estimativa da posição do robô. Uma vez que as todas as conexões são relativas apenas aos nós que as criaram, é possível integrar mapas de grandes áreas. Um aspecto indispensável durante a construção destas representações é analisar as incertezas do processo, com o objetivo de mitigar os erros dos mapas.

Além da representação lógica do ambiente, a maneira como os objetos do ambiente são representados é bastante importante. Geralmente essa caracterização se dá através de imagens das regiões de interesse. Dessa modo, reduz-se o problema de correspondência em mapas ao casamento de uma imagem em uma posição/orientação arbitrária à um modelo. De forma geral, a principal desvantagem deste tipo de método reside na grande necessidade de processamento, além da exigência de quantidade suficiente de características facilmente reconhecíveis para que a correspondência seja confiável.

2.3 REVISÃO SOBRE VISÃO COMPUTACIONAL

Devido ao interesse em sistemas de sensoriamento baseados em visão computacional, decidiu-se fazer uma breve revisão sobre o área. Primeiramente é abordada a parte de modelagem matemática de um arranjo estereoscópico de câmeras, baseando-se no modelo *pinhole* descrito por Stivanello (2008). Na sequência será abordado o problema de registro de imagens, em especial quanto a obtenção de características pontuais e os métodos usados no mapeamento das correspondências entre imagens.

2.3.1 Modelo pinhole

Para se poder apresentar a modelagem matemática de um arranjo estereoscópico, é necessário usar um modelo para representar o funcionamento das câmeras. Geralmente se usa o modelo *pinhole*, pois este é o modelo mais simples disponível (HARTLEY; ZISSERMAN, 2003).

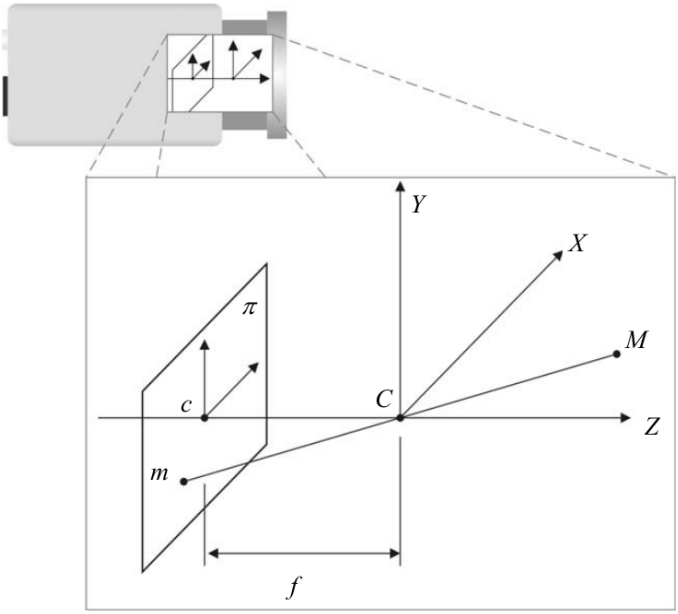


Figura 3 – Modelo *pinhole* (STIVANELLO, 2008).

Na figura 3, o modelo de câmera *pinhole* é apresentado, sendo composto por um plano de imagem π e por um ponto C que representa o centro de projeção da câmera. A distância entre π e C corresponde a distância focal f. A reta perpendicular ao plano π, que passa por C é representa o eixo óptico, e a intersecção c entre π e o eixo óptico é denominada como ponto principal. O conjunto de eixos coordenados 3-D cuja origem é C e para o qual o plano π é ortogonal ao eixo Z é conhecido como *quadro de referência da câmera*.

Desta forma, um ponto M no espaço com coordenadas $[X,Y,Z]^T$, é mapeado ao ponto m, o qual representa a intersecção do plano π com a reta que passa através dos pontos M e C. Por similaridade de triângulos, pode-se definir a transformação $[fX/Z,fY/Z,f]^T$, que mapeia as coordenadas do tridimensionais do mundo para coordenadas no plano de imagem da seguinte forma:

$$m = \left[f\frac{X}{Z}, f\frac{Y}{Z} \right]^T \tag{2.1}$$

A partir deste esquema, dois grupos de parâmetros devem ser definidos: (a) intrínsecos; e (b) extrínsecos. O primeiro grupo representa os parâmetros que descrevem as relações entre as coordenadas de pixel das imagens com um ponto no espaço referenciado no quadro de referência da câmera. O segundo grupo de parâmetros identificam a transformação entre o quadro de referência da câmera e o quadro de referência do mundo conhecido.

Fazem parte do conjunto de parâmetros intrínsecos a distância focal (f), as coordenadas do ponto principal c (c_x, c_y) e o tamanho efetivo do pixel (s_x, s_y). Além destes, distorções geométricas introduzidas por imperfeições óticas podem ser tratadas como distorções radiais, as quais podem ser definidas com a adição de outros dois parâmetros (k_1, k_2).

Já os parâmetros extrínsecos são definidos através de um vetor de translação t, e uma matriz 3×3 de rotação R. Assim um ponto W referenciado em um quadro de referência arbitrário do mundo, relaciona-se com o ponto M do quadro de referência da câmera segundo a equação $M = R(W - t)$.

A partir destes parâmetros, pode-se construir as equações que definem a projeção mundo-imagem, que mapeia um ponto 3D no ambiente em um ponto na imagem. Para facilitar os cálculos, utiliza-se uma notação matricial, juntando os parâmetros intrínsecos na matriz A e extrínsecos na matriz $[R|t]$, onde $R_n, n = 1,2,3$ representa os vetores equivalentes a cada linha da matriz R. Desconsidera-se aqui, eventuais distorções óticas.

$$A = \begin{bmatrix} -f/s_x & 0 & c_x \\ 0 & -f/s_x & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$[R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -R_1^T t \\ r_{21} & r_{22} & r_{23} & -R_2^T t \\ r_{31} & r_{32} & r_{33} & -R_3^T t \end{bmatrix}$$

Usa-se estas matrizes para definir a equação linear sobre o espaço homogêneo, onde $\tilde{W}$ representa o ponto tridimensional W, com a adição de um nova coordenada com valor unitário. Neste caso $\tilde{m}$ representa, em coordenadas homogêneas, a projeção de $\tilde{W}$ sobre o plano projetivo π.

$$\tilde{m} \cong A\,[R|t]\,\tilde{W} \tag{2.2}$$

Na equação 2.2, a matriz $P = A\,[R|t]$ é conhecida como matriz de projeção. A partir desta projeção, converte-se as coordenadas homogêneas $[u,v,w]^T$ de $\tilde{m}$ para coordenadas da imagem m, dividindo as duas primeiras coordenadas pela terceira, na forma $m = [u/w, v/w]$, equivalente ao resultado da equação 2.1.

2.3.2 Sistema Estereoscópico

Observando a equação 2.1 percebe-se que não é possível recuperar a informação tridimensional de um ponto m de uma imagem. Isso ocorre pois tal ponto pode corresponder à projeção de qualquer ponto sobre a reta que atravessa m e o centro de projeção C. Adicionando uma segunda câmera, torna-se possível a obtenção da informação necessária para a extração das coordenadas tridimensionais de um ponto que apareça em ambas as câmeras.

Para descrever o funcionamento de um sistema estereoscópico de câmeras, faz-se o uso da geometria epipolar. Segundo a geometria epipolar, dado um ponto em uma das imagens de um par estéreo, este deve aparecer sobre uma linha particular na outra imagem do par. A modelagem deste sistema é feita usando parâmetros intrínsecos e extrínsecos, assim como o modelo *pinhole* para uma câmera. Os parâmetros intrínsecos apenas agrupam os parâmetros intrínsecos de cada câmera. Já os parâmetros extrínsecos representam cada câmera do sistema em relação a uma referência. Desta forma as matrizes de rotação, e os vetores de translação R_e, R_d, t_e e t_d relativos a esta referência são combinados, de forma a definir a matriz de rotação R e o vetor

de translação t para o sistema na seguinte forma:

$$R = R_e R_d^T$$
$$t = t_e - R^T t_d$$

A figura 4 apresenta o diagrama de um sistema estéreo genérico, formado por duas câmeras *pinhole*. Neste diagrama estão representados os centros de projeção C_e e C_d das câmeras esquerda e direita, e os respectivos planos de imagem π_e e π_d. Os vetores $m_e = [u_e, v_e, w_e]$ e $m_d = [u_d, v_d, w_d]$ representam a projeção do ponto M em cada um dos planos de imagem. Os pontos e_e e e_d são os epipolos do sistema estéreo. Eles representam a projeção do centro de projeção de uma câmera no plano de imagem da outra. Assim, o epipolo e_e é a imagem de C_d em π_e, e vice-versa.

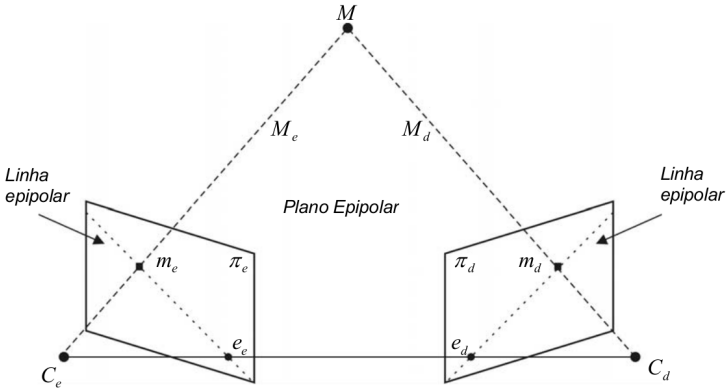


Figura 4 – Geometria epipolar de um sistema estereoscópico (STIVANELLO, 2008).

Os pontos M, C_e e C_d definem o plano epipolar, e a intersecção deste com os planos de imagem π_e d π_d define as linhas epipolares. Dado um ponto em um plano de imagem e o centro de projeção deste plano, a reta que atravessa estes dois pontos aparece no plano de imagem da outra câmara como uma linha epipolar. Ela recebe este nome, pois todas as linhas epipolares intersectam-se nos epipolos.

Desta propriedade surge a *restrição epipolar*. Dado m_e, M pode estar em qualquer posição na linha M_e que liga C_e e m_e. Desta forma, o ponto m_d deve pertencer a linha epipolar que representa a projeção de M_e sobre o plano π_d, estabelecendo assim a relação entre os pontos das duas imagens. Con-

forme Stivanello (2008), a restrição epipolar pode ser definida algebricamente por meio da equação 2.3, caso os parâmetros intrínsecos sejam conhecidos:

$$\hat{m}_d^T (t \times R\hat{m}_e) = 0 \;:=\; \hat{m}_d^T [t]_\times R\hat{m}_e = 0 \tag{2.3}$$

onde $[t]_\times$ representa a matriz anti-simétrica do vetor de translação t e $\hat{m}_e$ e $\hat{m}_d$ representam os pontos nas respectivas imagens em coordenadas normalizadas.

A matriz $E = [t]_\times R$ é conhecida como *matriz essencial*, que descreve a relação entre as coordenadas normalizadas, possuí cinco graus de liberdade, correspondendo a uma rotação e uma translação 3-D. Caso não se tenha os parâmetros intrínsecos das câmeras, usa-se a *matriz fundamental* F, equivalente a:

$$F = A_d^T E A_e^{-1}$$

onde A_d e A_e são as matrizes de parâmetros intrínsecos de ambas as câmeras. A matriz fundamental relaciona os pontos de ambas as imagens em coordenadas de pixel na forma $m_d^T F m_e = 0$. Usando a matriz fundamental pode-se definir diretamente as linhas epipolares l_e e l_e da seguinte maneira:

$$l_d = F m_e$$
$$l_e = F^T m_d$$

Para se obter as matrizes fundamental e essencial é necessário usar o procedimento de calibração não abordado neste texto. Mais informações sobre este tema podem ser encontradas em Stivanello (2008) e Hartley e Zisserman (2003).

2.3.3 Registro de imagens

O registro de imagens é o processo que alinha espacialmente duas ou mais imagens de uma cena (GOSHTASBY, 2005), sendo indispensável em aplicações que requerem a análise de imagens. Em especial, o registro de imagens permite o reconhecimento de objetos de uma imagem, caso sejam fornecidas imagens para comparação, as quais se conheça a identidade dos objetos que estas contém. Em Goshtasby (2005) verifica-se que o processo de registro de imagens é formado por duas etapas principais: a) seleção de características; b) determinação do mapeamento entre as características das imagens.

a) Seleção de características

Para que se possa aplicar o processo de registro de imagens é necessário que características das imagem sejam selecionadas, para que seja feita a correspondência entre imagens. As características utilizadas no registro de imagens podem ser pontos, linhas, curvas, padrões de textura regiões das imagens e etc.

Os pontos são o tipo de característica mais desejado, devido a maior facilidade em se obter a transformação que mapeia as coordenadas dos pontos de múltiplas imagens. Geralmente este tipo de característica é conhecido como **pontos de interesse**, **referências**, ou **pontos de controle**.

Vários características de imagens podem ser tratadas como pontos, sendo que cantos e vértices são exemplos mais comuns. Porém, há muitos objetos que não são bem descritos por usando esse tipo de característica. Com isso, tem havido interesse no desenvolvimento de descritores de ponto de interesse, que consigam identificar regiões com características significativas que sejam identificáveis sob condições variáveis. Descritores que consigam ser imunes a algum tipo de transformação são chamados **invariantes** a tal condição.

Entre os vários descritores disponíveis atualmente através de bibliotecas de visão computacional[2], destaca-se o SIFT (*Scale-Invariant Feature Transform*), o qual se mostrou bastante útil na área de movimentação de robôs, como demonstrado na seção 2.4.2. Para descrever imagens este algoritmo busca pontos que sejam invariantes a variação de escala, através de um processo de simulação (MOREL, 2009) e os descreve construindo um histograma indexado tridimensionalmente da região em torno do ponto. Desta forma obtém-se um descritor invariante a transformação de escala, translação e rotação no plano de imagem, tendo também certa robustez quanto a mudanças de perspectiva. Uma explicação mais detalhada será apresentada no capítulo 4.

b) Determinação do mapeamento

Com o conhecimento das coordenadas do conjunto de pontos correspondentes entre as imagens, pode-se determinar a transformação que mapeia a imagem que está sendo avaliada com relação a geometria da imagem referência. O tipo de transformação a ser usado depende da diferença geométrica

[2] A biblioteca openCV implementa vários descritores. Mais informações podem ser encontradas em:
`http://docs.opencv.org/modules/features2d/doc/feature_detection_and_description.html`
`http://docs.opencv.org/modules/nonfree/doc/feature_detection.html`

entre as imagens avaliadas, da exatidão da correspondência entre as características avaliadas e da densidade destas correspondências.

A presença de ruídos e inconsistências nas correspondências faz com que métodos de ajuste (*fitting*) e agrupamento (*clustering*) geralmente estejam associados a definição do mapeamento entre imagens. Os métodos de ajuste, como mínimos-quadrados, estão associados a correção de erros que possam ser modelados como ruídos com média nula (GOSHTASBY, 2005).

Os métodos de agrupamento são necessários quando existe uma quantidade considerável de falsas correspondências. Quando há disponibilidade de uma alta densidade de correspondências, é possível utilizar de métodos iterativos para que os parâmetros da transformação sejam definidos de forma mais robusta usando apenas parte do conjunto de correspondências. O RANSAC[3] é um método não-determinístico bastante utilizado quando a presença de falsos-positivos entre as correspondências é inferior a 50%.

Quando a presença de falsos-positivos é significativa, a transformada generalizada de Hough (BALLARD, 1981) é utilizada. Como o nome indica, este algoritmo é a generalização do trabalho de Paul Hough[4], que era um método de extração de características, que originalmente obtinha a descrição de linhas através de seus parâmetros algébricos. A transformada de Hough caracteriza-se pela definição de um espaço de parâmetros (espaço de Hough) definido em função do tipo de característica que se busca.

De uma maneira simplificada, pode explicar o funcionamento da transformada de Hough para o caso de retas da seguinte maneira: Dada uma imagem Im definida sobre plano cartesiano com eixos X e Y, a qual foi pré-processada de forma a permanecerem apenas os pixels não nulos que pertençam a arestas, busca-se por retas definidas por dois parâmetros: a inclinação $\mathbf{A}$ e o ponto $\mathbf{B}$ na qual a reta atravessa o eixo Y na equação 2.4.

$$y = \mathbf{A}x + \mathbf{B} \tag{2.4}$$

Desta forma o espaço de Hough para busca por retas é formado por um plano com todas as possíveis inclinações e posições onde se pode cruzar o eixo Y.

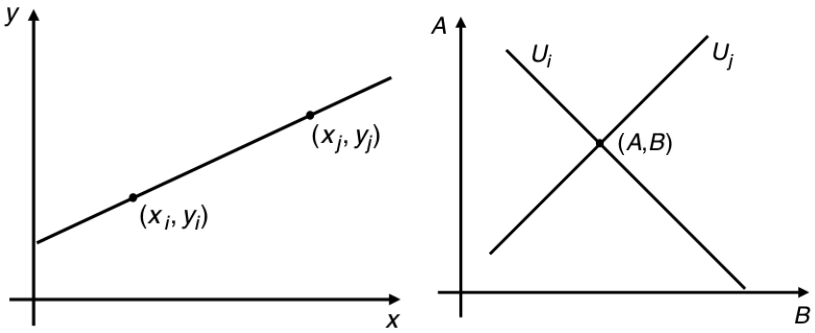
Assim, para cada ponto $p_k = (x_k, y_k)$ pertencente a imagem Im, verifica-se todas as retas que possivelmente passariam por p_k. Reorganizando a equação 2.4 em função dos parâmetros $\mathbf{A}$ e $\mathbf{B}$ e usando as coordenadas de p_k,

[3]Fischler,M.; Bolles,R. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. Comm. of the ACM **24** (6), 1981, 381–395

[4]Hough,P. *Machine Analysis of Bubble Chamber Pictures*, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

$$\mathbf{A} = \frac{y_k}{x_k} - \mathbf{B}\frac{1}{x_k} \Rightarrow \mathbf{A} = u\mathbf{B} + v \mid u = -\frac{1}{x_k} \; e \; v = \frac{y_k}{x_k} \tag{2.5}$$

verifica-se que todas as retas que passam por p_k definem retas U_k no espaço de Hough parametrizadas por u e v, conforme verificado na figura 5.



(a) Imagem contendo uma linha (b) Espaço de Hough equivalente

Figura 5 – Tranformada de Hough para linhas (NIXON; AGUADO, 2008)

Após traçar as retas U_k para todos os pontos p_k, é possível identificar as retas presentes na imagem Im observando o ponto do espaço de parâmetros onde o cruzamento entre as retas U_k e mais intenso. Quanto mais retas cruzarem em um determinado ponto (A,B) do espaço de parâmetros, maior será a evidência da existência de uma reta na imagem Im a qual é definida pelos parâmetros A e B.

Para outros identificar outros tipos de objeto através da transformada de Hough, basta criar uma parametrização adequada. O exemplo demonstrado é apenas didático, sendo que exitem modelagens mais eficientes. Uma das otimizações mais comuns é a utilização da informação da derivada em cada ponto p_k de forma a reduzir a quantidade de possíveis retas que passam por tal ponto. As implementações da transformada de Hough geralmente fazem uso da discretização do espaço de Hough como forma de torná-la mais eficiente. Este artifício também permite que as correspondências usadas para busca de parâmetros sejam agrupadas em conjuntos que concordem quanto a definição dos parâmetros. Correspondências falsas tendem a gerar parâmetros aleatórios, que não se acumulam em uma determinada região do espaço de parâmetros, tornando a transformada de Hough significativamente robusta a presença de falsos-positivos no conjunto de correspondências.

Quanto as transformações propriamente ditas, estas podem ser lineares ou não-lineares. Transformações não-lineares são mais indicadas quando se conhece razoavelmente a origem das diferenças geométricas, como no caso de objetos não-rígidos, e por isso não serão abordadas neste trabalho. No contexto desta dissertação foram usadas para definir o mapeamento geométrico de correspondências entre imagens as transformações de similaridade e afim. A primeira é mais restritiva e permite apenas três variações sobre as imagens: escala, rotação e translação. Já a transformação afim permite ainda operações de cisalhamento.

2.4 MOVIMENTAÇÃO ATRAVÉS DE VISÃO COMPUTACIONAL

Dentre os vários métodos de sensoriamento, a visão computacional tem sido alvo de inúmeras pesquisas, visto que esta pode aumentar o âmbito de aplicação dos robôs (BONIN-FONT et al., 2008). A literatura mostra que múltiplas configurações de câmeras têm sido usadas, das quais os sistemas monoculares e os binoculares (estéreo) são os mais comuns. Sistemas trinoculares tem ganhado destaque devido a maior robustez, consequência do maior número de restrições geométricas, que reduzem a aplicação de heurísticas na definição da correspondência entre as imagens estéreo (AYACHE; LUSTMAN, 1991). Outro sistema que tem ganhado destaque, são os sistemas com câmeras omnidirecionais, devido a possibilidade de se ter um uma visão de 360°.

Dois trabalhos apresentaram uma revisão de estudos com aplicação de técnicas de visão computacional ao campo da robótica móvel. Desouza e Kak (2002) apresentaram o compêndio das pesquisas desenvolvidas até o fim dos anos noventa. Bonin-Font et al. (2008) complementaram o trabalho anterior, adicionando os desenvolvimentos feitos durante o início dos anos 2000. Em ambos utilizou-se uma classificação semelhante aquela apresentada na seção 2.2.2.

2.4.1 Aplicações baseadas em visão computacional

A principal vantagem de sistemas baseados em visão computacional está na grande quantidade de informação que estes disponibilizam. Porém, o processamento necessário para a compreensão e conversão desta informação em dados úteis é bastante elevada. De forma geral, os sistemas baseados em visão utilizam algum tipo de representação simplificada, o qual foca em um tipo específico de informação.

Santos-Victor et al. (apud DESOUZA; KAK, 2002) desenvolveram um sistema de movimentação através de estimativas baseado em fluxo ótico. Este sistema mimetiza o comportamento visual de abelhas, utilizando duas câmeras em um arranjo divergente. Devido à maneira como são posicionados os olhos em insetos, seu campo de visão binocular é extremamente estreito, o que dificulta a obtenção de informações de profundidade. Por isso, acredita-se que estes animais utilizam mecanismos de movimentação que privilegiam informações derivadas de movimentos em detrimento a informações de profundidade. Desta forma, se o robô está se deslocando em linha reta através do centro de um corredor, então a diferença entre a velocidade entre os quadros da câmera da esquerda e direita será próxima de zero. Porém, se a velocidade for diferente, então o robô estará se deslocando em direção ao lado o qual a velocidade entre os quadros de imagem é menor. Assim, com relação a implementação do sistema, a ideia básica é mensurar a velocidade entre os quadros de cada câmera, compará-los de forma a identificar a direção do movimento.

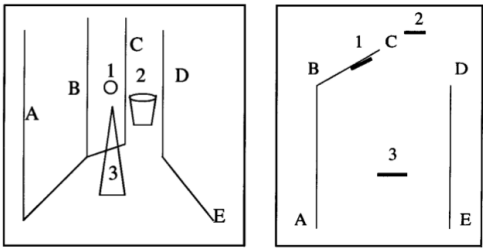
Uma característica interessante desta técnica, é que ao ser utilizada em um robô diferencial, ela tem acoplamento direto com a cinemática do mesmo. Mas, o principal interesse neste trabalho ocorreu pela abordagem bastante incomum, ao utilizar apenas a dinâmica da imagem em detrimento da informação da mesma. Isto demonstra a grande quantidade de soluções para o problema de movimentação de robôs.

Nister et al. (2004) desenvolveram um método com propósito de obter estimativas de movimentação de maneira robusta a partir de um tipo específico de informação. A técnica conhecida como odometria visual é baseada no rastreamento quadro a quadro das informações que caracterizam cada imagem. Além disso, há variações tanto para sistemas monoculares quanto estéreo. Neste sistema utiliza-se na fase de detecção de características, o detector de cantos de Harris. A correspondência é feita entre pares de quadros, limitando a disparidade entre as características equivalentes dependendo da velocidade do robô. Além disso, a equivalência entre pares de características só é aceita caso esta seja consistente. Por consistência entende-se que para cada par de características correspondentes, seus elementos sejam mutuamente a melhor correspondência possível.

A robustez na fase de estimação do movimento é obtida através de consenso entre os múltiplos pares correspondentes por meio do algoritmo RANSAC (*Random Sample Consensus*) e múltiplas iterações para refinamento da hipótese. Além disso usa-se *firewalls* para limitar a propagação de erros. Os *firewalls* são pontos de controle para os quais as estimativas subsequentes não se baseiam em observações feitas anteriormente ao mesmo. Como resultado é obtido um sistema bastante efetivo, com erro inferior a 5%, mesmo em situações desfavoráveis como a movimentação em prados, os quais dificultam

a obtenção de cantos para o rastreamento. O interesse neste trabalho ocorreu por ser uma aplicação terrestre do sistema feito por Maimone et al. (2007), que demonstra o desempenho de técnicas baseadas em detectores locais (detector de cantos de Harris) na estimação de poses.

Com relação aos objetivos desta dissertação, a abordagem de Kim e Nevatia (1999) representa uma referência fundamental. Em uma abordagem híbrida, estes pesquisadores criaram um sistema baseado em mapas e marcações naturais, as quais representam itens de interesse no ambiente. Como diferencial, os autores propuseram um sistema de movimentação através de navegação simbólica. Neste tipo de sistema, os planos de navegação são feitos em linguagem formal, próxima a usada por humanos: "vá através da `porta` à esquerda da primeira `mesa` à direita". Neste caso, `porta` e `mesa` são símbolos que representam marcações naturais do ambiente. O sensoriamento neste sistema é feito através de um arranjo trinocular na configuração de triângulo equilátero. A representação do ambiente para fins de navegação é feita em duas partes. A parte referente à representação espacial tem como objetivo representar obstáculos de maneira robusta. Por outro lado, a parte de representação genérica tem a finalidade de reconhecer objetos comuns em ambientes internos, com o objetivo de utilizá-los como referência para navegação.



(a) Segmentos de reta (b) s-map

Figura 6 – Representação espacial de Kim e Nevatia (1999)

Para representar obstáculos como paredes e objetos não identificáveis pela representação genérica, os autores propuseram a utilização do *s-map* (figura 6b) . Esta representação tenta comprimir (*squeeze* em inglês) as informações tridimensionais do ambiente, obtidas através das câmeras, em um espaço bidimensional. Para isso, esta técnica baseia-se em uma série de hipóteses sobre o ambiente e o comportamento visual de objetos, de acordo com o tamanho destes com relação ao robô. A partir destas hipóteses, das restrições geométricas do arranjo trinocular e dos segmentos de reta obtidos do mesmo, este algoritmo constrói superfícies para representar os obstáculos (paredes e objetos) e criar o *s-map* (figura 6).

No contexto deste trabalho a parte referente a representação genérica é mais importante. A importância se dá no sentido de permitir a identificação de objetos, que são os símbolos (e marcas) para executar a navegação. Para fazer o reconhecimento, os autores criaram um método baseado na funcionalidade dos objetos. Nesta abordagem, cada objeto é descrito por suas **superfícies significantes** e sua **evidência funcional**. As superfícies significantes são escolhidas de acordo com o papel que esta tem na função do objeto. No caso de uma mesa por exemplo, esta é descrita por uma "área de trabalho"que representa o tampo, e alguma superfície que faz parte da estrutura de suporte, isto é, as pernas. As evidências funcionais surgem a partir da utilização do objeto no contexto de sua função. A observação de itens sobre um tampo constitui indício de que tal superfície pertence a uma mesa. Para fins de navegação, os autores utilizaram mesas e portas como marcações naturais. A descrição destes objetos dentro do sistema é apresentada na tabela 2.

O reconhecimento efetivo parte da detecção das superfícies significantes e evidências funcionais nos segmentos de reta obtidos na parte de representação espacial, utilizando tanto informações bidimensionais quanto tridimensionais. Além disso, assume-se que os objetos aparecem em uma "pose padrão", isto é, no modo que estes geralmente estão dispostos no ambiente. A detecção de superfícies significantes se dá de dois modos: através de restrições quanto as primitivas que descrevem as mesmas, ou restrições no relacionamento com outras superfícies significantes. Cada superfície significante é descrita por quatro primitivas: orientação, variação de altura, forma primária e tamanho. As duas primeiras são utilizadas para diminuir o espaço de busca dos segmentos de reta candidatos a formarem uma superfície significante.

Tabela 2 – Base de dados dos objetos a partir de características funcionais. (KIM; NEVATIA, 1999)

Objeto	**Representação funcional**	**Superfície significante**	**Evidencia funcional**
Mesa	Trabalhando numa altura confortável	Área de trabalho	Materiais sobre a área de trabalho
	Superfície de Armazenamento	Suportes	
Porta	Passagem	Quadro atravessável	Materiais visíveis através da porta
	Fechamento	Painéis da Porta	

A duas últimas primitivas são usadas na organização perceptiva (*Perceptual Grouping*) dos segmentos de reta candidatos. A tabela 3 apresenta exemplos da utilização destas primitivas para descrever superfícies significantes.

A primitiva de forma primária identifica apenas a forma mais comum de tal superfície significativa. No caso de uma mesa, que geralmente é retangular, a utilização apenas das primitivas impediria o reconhecimento de uma mesa com tampo redondo. Neste caso o relacionamento com as outras superfícies significativas tem ação preponderante: o sistema reconhece-a encontrando uma forma arbitrária, não apenas utilizando as outras três primitivas, mas também com a ajuda de outras superfícies, como aquelas das pernas da mesa.

Para detectar objetos através de evidências funcionais são necessárias duas etapas. Inicialmente é preciso encontrar as superfícies significativas capazes de gerar evidências funcionais compatíveis com os objetos a serem identificados. Depois encontra-se segmentos dentro do domínio das superfícies identificadas na fase anterior, os quais possuem informação tridimensional compatíveis com evidências funcionais disponíveis. Os segmentos que atenderem a este critério tornam-se as evidências funcionais. No caso de uma porta, os segmentos de reta que estejam ao mesmo tempo dentro do quadro da porta e atrás do mesmo, constituem evidências funcionais da existência de uma porta.

Apesar deste sistema de reconhecimento de objetos atender suficientemente os propósitos dos autores do sistema ,há de serem consideradas sua limitações, principalmente quando envolvem um conjunto de objetos complexos. Os próprios autores suscitaram a possibilidade de falso-positivos derivados de múltiplas classes de objetos possuírem superfícies significativas e evidências funcionais parecidas. Além disso, a premissa da pose padrão não

Tabela 3 – Descrição de superfícies significantes através de primitivas. (KIM; NEVATIA, 1999)

Primitiva	**Área de trabalho**	**Quadro atravessável**
Orientação	Horizontal	Vertical
Variação de Altura	70-120cm	chão - 2.2m
Forma Primária	Retângulo	Retângulo
Tamanho	Largura: 1-2m Profundidade: 0.7-2m	Largura: 0.7-2.5m Profundidade: 1.8-2.5m

é útil quando se busca objetos que não possuem essa propriedade. Essas limitações tornaram necessária uma pesquisa sobre métodos alternativos para o reconhecimento de objetos.

2.4.2 Descritores locais e reconhecimento de objetos

Recentemente, métodos baseados em descritores locais de pontos de interesse têm tido destaque na comunidade científica. Desenvolvimentos recentes tornaram este tipo de característica invariante a transformações de imagens, além das já conhecidas propriedades de robustez a oclusão e distintividade (MIKOLAJCZYK; SCHMID, 2005). O grande sucesso na utilização deste tipo de técnica em aplicações de reconhecimento de objetos, recuperação de imagens, tarefas de localização em robôs etc, levou diversos pesquisadores a se dedicarem à comparação destes descritores.

No âmbito da robótica móvel, Klippenstein e Zhang (2007) avaliaram três descritores locais aplicados a tarefa de mapeamento e localização simultânea (SLAM): (i) SIFT (*Scale-Invariant Feature Transform*); (ii) Detector de cantos de Harris; e (iii) as características utilizadas pelo rastreador KLT (Kanade-Lucas-Tomasi). Além disso, os autores também avaliaram três métodos de correspondência: (i) Vizinho-mais-Próximo (*Nearest-Neighbor*) baseado na razão entre distâncias; (ii) correlação cruzada normalizada; e (iii) rastreador KLT. Os testes foram feitos sobre dois conjuntos de imagens gerados à partir de uma câmera em um robô. Este capturava imagens a cada $150\,mm$ de deslocamento ou 5° de rotação. As diferenças entre os conjuntos foram quanto ao ambiente e a distância percorrida: 30 metros para um ambiente tipo escritório e 10 metros para um ambiente de pesquisa com robôs. Além de fazer a avaliação com toda a sequência de imagens, também foi feita utilizando sub-amostragens, com o intuito de simular deslocamentos e rotações de maior amplitude.

A avaliação das correspondências entre os descritores de características locais foi feita através da homografia entre par de imagens de interesse, os quais são sequenciais no conjunto. Uma correspondência entre características locais só é considerada verdadeira, quando o ponto correspondente de uma imagem na outra estiver, no máximo, a dois pixeis da linha epipolar gerada pela matriz fundamental entre as imagens.

Para mensurar o desempenho na busca por correspondências, foram utilizadas as métricas de precisão e recuperação. Como resultado, os pesquisadores afirmaram que todos os descritores têm um funcionamento adequado usando a amostragem de imagens mais alta. Neste caso, o KLT tem o melhor desempenho devido à capacidade de rastreamento através de múltiplas ima-

gens. Porém, ao aumentar a amplitude de movimento entre as imagens, ou seja diminuindo a taxa de amostragem, o SIFT e Harris passam a ter vantagem, principalmente o SIFT no caso de maior variação de orientação.

Ainda na área da robótica, destaca-se o trabalho de Ramisa et al. (2009), que avaliou o método de reconhecimento de objetos proposto por Lowe (2004)[5]. Os autores compararam o desempenho da técnica, fazendo modificações em várias partes da mesma, para poder testar múltiplos extratores de características. Os testes foram feitos com uma base de dados própria, que tinha 29 objetos representativos ao problema de movimentação de robôs. As imagens foram obtidas diretamente de um robô enquanto este navegava no ambiente do laboratório de pesquisa. O ambiente não sofreu nenhuma modificação, o que fez com que as imagens de teste fossem afetadas por variações de iluminação, borramentos, grande variação de escala e oclusão.

Para comparar as diferentes alternativas de utilização do método de Lowe, os autores utilizaram a medida `f-measure`. Esta medida é comum na área de reconhecimento de padrões, e é utilizada para mensurar a relação entre a precisão e a recuperação do método testado. Como resultado os pesquisadores concluíram que, considerando a base de dados com todas as possíveis degradações das imagens e as modificações na proposta original, é possível detectar aproximadamente 60% dos objetos bem texturizados, com uma precisão de 90%. Além disso, eles consideraram plausível a utilização desta técnica uma vez que a maioria das configurações testadas tiveram um tempo de execução inferior a um segundo.

Pesquisou-se também o uso do SIFT em aplicações robóticas. Vários trabalhos foram encontrados, dos quais dois foram selecionados para serem apresentados na sequência. Chen e Tsai (2010) desenvolveram um sistema para patrulhamento de ambientes internos, no qual o robô se localiza por meio de características locais de objetos planares fixos ao longo do caminho, objetos estes que devem ser monitorados. Desta forma, o robô tem a tarefa de cobrir o ambiente, localizando-se através de objetos. Considerando as classificações da seção 2.2.1, este é um sistema de movimentação com foco na interação com outros objetos, por isso é classificado quanto a escala, como local. A dinâmica deste sistema é composta por dois estágios: **aprendizado** da trajetória e **navegação** pela mesma.

Na etapa de aprendizado, o robô é conduzido manualmente pelo caminho desejado, aprendendo não só a trajetória desejada (em coordenadas métricas e/ou angulares), mas também as informações relevantes sobre os objetos que devem ser monitorados. Para isso, é utilizada uma câmera PTZ (*pan-tilt-zoom*) que obtêm imagens dos objetos. Estas imagens são associadas a nós no mapa de navegação, que efetivamente transformam os objetos

[5] Baseado no descritor SIFT. Será abordado no capítulo 4.

em marcações do ambiente. Na etapa de navegação o robô percorre a trajetória adquirida na fase anterior e monitora os nós, os quais contêm os objetos de interesse. Considerando a classificação da seção 2.2.2 este é um sistema baseado em mapas geométricos para decidir as ações, mas que usa marcações para fazer a localização.

As incertezas ao reproduzir os deslocamentos métricos aprendidos durante o aprendizado, faz com que ocorram divergências nas trajetórias percorridas em cada fase, e por sua vez diferenças nas poses das quais as imagens são obtidas. Isso provoca uma mudanças significativa na aparência das imagens que devem ser comparadas. Para sobrepujar este problema, os pesquisadores utilizaram o SIFT como técnica para testar a correspondência entre imagens. Porém, a maior contribuição deste trabalho está na utilização das características SIFT em uma estratégia para correção da trajetória.

A partir das correspondências entre as características SIFT das imagens da etapa de aprendizado e navegação é encontrada a transformação afim T, que define o mapeamento relativo entre as imagens. Isto é feito de acordo com a proposta feita por Lowe (2004). Aplicando T nas linhas de calibração l_L, previamente definidas na imagem de aprendizado (seja de maneira natural ou artificial), se obtém a linha de calibração l_N da imagem de navegação. Utilizando um sistema de coordenadas de referência e aplicando uma transformação tridimensional sobre as linhas l_L e l_N se obtém as poses nas quais as imagens foram obtidas com relação ao sistema de coordenadas. Então a partir das poses, deriva-se a translação e a rotação necessárias para corrigir a posição.

Para testar os sistema foram usados os corredores de um edifício (Figura 7) cujo comprimento totalizou 75,83 metros. Neste percurso, nove objetos foram selecionados para serem monitorados durante a locomoção. A distância média entre estes objetos era de 7.58 metros. O erro de localização dos objetos foi sempre inferior a 5% para os deslocamentos e 2° para a orientação. Com relação a execução das trajetórias entre os nó, o erro médio ao final da execução foi de 2,6 cm para os deslocamentos e 1.2° para a orientação.

No teste de correção de trajetória, o robô foi posicionado fora do caminho, fazendo-o seguir até o nó mais próximo. Neste caso o erro máximo foi de aproximadamente 6% para deslocamentos e $2,4^\circ$ para orientação. Segundo os autores, o erro médio do sistema é comparável a outros métodos concorrentes. A vantagem deste método é poder movimentar-se utilizando objetos planares como marcação com um erro mínimo. A desvantagem está na dificuldade de aplicar esta técnica para objetos com formas mais complexas.

Miro et al. (2006) desenvolveram um sistema SLAM de mapeamento

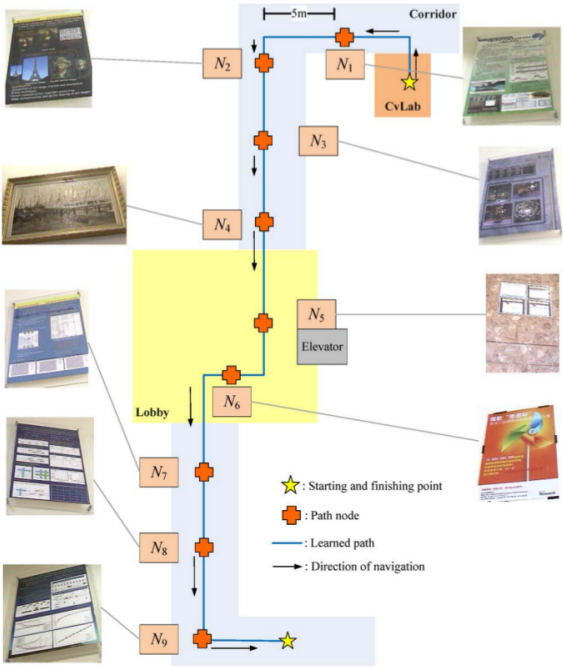


Figura 7 – Mapa de navegação (CHEN; TSAI, 2010)

e localização simultânea através de características SIFT, mapas de profundidade (*depth maps*) e filtro de Kalman estendido. O sensoriamento para a obtenção dos mapas de profundidade e as características SIFT foi feito através de arranjo estéreo, usando lentes grande-angular. Segundo os autores, este tipo de objetiva tem melhores resultados, pois geram uma maior quantidade de características SIFT. Com foco em grandes ambientes fechados, tal trabalho teve preocupações quanto ao funcionamento de técnicas baseadas em SIFT em escalas locais e globais.

Os autores citam o fato de ambientes internos possuírem muitas regiões as quais são muito similares, e que por isso, há grandes chances de existir erros na correspondência de características SIFT. Para solucionar este problema eles fazem uso do *Bayesian Innovation Gate* no estimador do filtro de Kalman. Outro ponto importante do trabalho está na utilização da arquitetura Player para fazer a interface com a plataforma robótica. A partir do Player foi feita a aquisição síncrona dos dados dos sensores e controle do robô.

Os autores efetuaram três tipos de testes. O primeiro foi feito em um escritório com área aproximada de $36 m^2$. O segundo teste foi desenvolvido em uma arena não estruturada para experimentação de problemas de busca e resgate, com aproximadamente $49 m^2$. O último foi desenvolvido em um grande ambiente, com um deslocamento total da ordem de $150 m$. Em todos os experimentos o método de SLAM visual baseado no SIFT foi claramente superior a métodos baseados puramente em odometria.

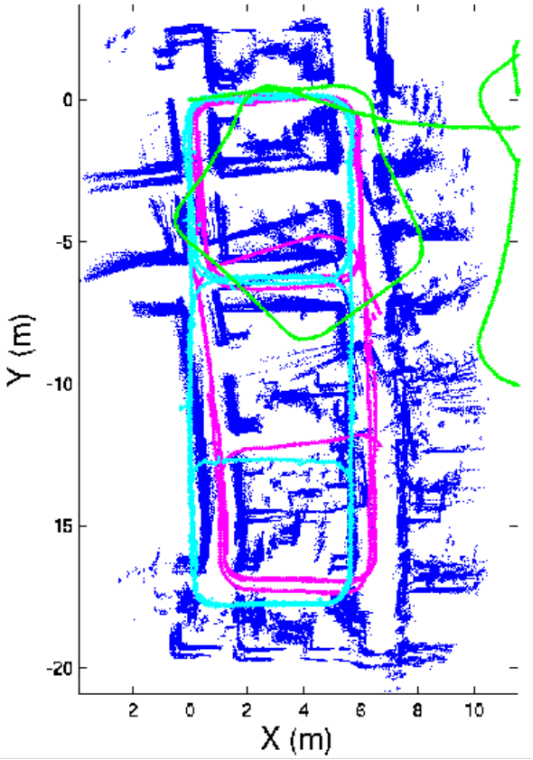


Figura 8 – Movimentação em grande ambiente (MIRO et al., 2006). Em verde a estimativa fornecida pela odometria. Em magenta a estimativa oriunda do método SLAM. Em ciano a movimentação real provida por *laser*. Em azul, as paredes obtidas da superposição do laser sobre as poses do SLAM.

O resultado mais importante deste trabalho foi obtido no último teste. Com o grande deslocamento feito neste teste foi percebida uma degradação na qualidade da localização, isto é, as posições oriundas do sistema SLAM eram

inconsistentes com as leituras reais do ambiente. Na figura 8 é possível ver a diferença entre a movimentação real, provida por *laser*, e as estimativas do método SLAM (em magenta na figura). Além disso, fica nítida a significativa inferioridade dos resultados obtidos a partir de odometria pura (em verde).

Os estudos realizados neste capítulo revelaram o grande espectro de soluções ao problema de movimentação robótica, bem como as possíveis classificações que estes sistemas podem receber, seja em função da escala da tarefa, seja quanto ao tipo de estratégia a ser utilizada. Dentre os métodos de sensoriamento atuais destacou-se a visão computacional, devido a grande flexibilidade e a quantidade de informação que esta fornece.

O bom desempenho do SIFT nas avaliações e nas aplicações em robótica levaram a decisão de aprofundar o estudo do método de reconhecimento de objetos baseado nele. Porém, antes deste estudo é necessário desenvolver a estrutura para que possam ser feitos os testes deste sistema. Desta forma, o próximo capítulo trata do desenvolvimento do ambiente de testes para aplicações de visão em robótica móvel através de simulação.

3 CONSTRUÇÃO DO AMBIENTE DE TESTES

Na revisão bibliografia apresentada no capítulo 2, observou-se que os pesquisadores geralmente utilizaram conjuntos de imagens obtidos a partir de robôs ou disponíveis na internet. Porém, no contexto deste trabalho, onde se busca implementar algoritmos de visão computacional, é de primordial importância a capacidade de testar as implementações sob múltiplas condições.

Produzir estas condições significa controlar precisamente a posição e configuração das câmeras, objetos e iluminação. As dificuldades relacionadas ao controle de todos estes parâmetros dificulta a comparação de técnicas e de resultados obtidos com parâmetros diferentes. Assim, a utilização de ferramentas capazes de simular ambientes tridimensionais com alta fidelidade representa uma grande vantagem, seja com relação ao tempo de desenvolvimento, quanto a variedade de testes possíveis.

Este capitulo apresentará inicialmente uma avaliação de ambientes de desenvolvimento para robótica (ADRs) através da metodologia específica para este tipo de ferramenta. Na sequência será abordada a construção de um simulador, o qual foi implementado através de um *framework*, desenvolvido com base nas ideias encontradas durante o estudo das ferramentas disponíveis. Serão abordados aspectos relacionados à parte gráfica e ao uso e criação do *framework* e do simulador implementado sobre este.

3.1 AVALIAÇÃO DE AMBIENTES DE DESENVOLVIMENTO PARA ROBÓTICA

Por meio da pesquisa bibliográfica encontrou-se uma grande quantidade de ADRs. Vários exemplos foram apresentados nos trabalhos de Harris e Conrad (2011) e Castillo-Pizarro et al. (2010). Para avaliar tais ferramentas, decidiu-se adaptar a metodologia proposta por Kramer e Scheutz (2007), que avaliaram nove ADRs.

Esta metodologia é abrangente, avaliando desde formalismo da modelagem proposta pelas ferramentas até o impacto que estas causaram. Neste contexto é necessário ter em vista que o objetivo é desenvolver e testar uma aplicação de visão computacional, a qual representa apenas um componente e não um sistema robótico completo. Devido a este fato, e a amplitude da metodologia de Kramer e Scheutz (2007), que avalia as várias fases e necessidades do desenvolvimento de um robô, foram considerados apenas os critérios que têm relação direta com o desenvolvimento de sistemas de visão para robótica. A tabela 4 resume os critérios que foram utilizados neste trabalho, não

necessariamente executados nesta ordem.

Tabela 4 – Critérios de avaliação de ADRs derivados da metodologia de Kramer e Scheutz (2007).

Classe	**Subclasse**	**Critérios**
Funcionalidade	Especificação	Engenharia de Software
	Suporte a plataformas	Simulação[a]
	Infraestrutura	Registro/Depuração Comunicação de baixo nível
	Implementação	Linguagem de Programação Documentação
Usabilidade	Instalação	Documentação Componentes externos Próprio ADR Experiência
	Usabilidade Básica	Implementação de teste Criação do ambiente[b] Qualidade gráfica[b] Experiência
Influência		Utilização Compatibilidade

[a]Pré-requisito para avaliação
[b]Critérios próprios

3.1.1 Metodologia

Em sua metodologia, Kramer e Scheutz (2007) usaram três classes de critérios: **Funcionalidade**, **Usabilidade** e **Influência**. A primeira se refere aos recursos que o ADR disponibiliza seja na fase de projeto e implementação de um sistema robótico. A segunda classe refere-se à facilidade na utilização de tal ferramenta, e a última refere-se ao impacto que a ferramenta teve na comunidade científica.

a) Critérios Funcionais

Originalmente, os critérios funcionais apresentados pelos autores da metodologia estavam ligados principalmente à especificação e à implementação de arquiteturas de software, o suporte à plataformas de hardware e aspectos de computação distribuída. Considerando o sistema de visão como um módulo do software de um robô, este deve ser independente de arquiteturas ou plataformas robóticas. Isto torna muitos dos critérios sugeridos pelos autores irrelevantes neste contexto específico, e por isso, foram desconsiderados. Mesmo assim, optou-se por apresentar estes critérios, justificando sua não utilização. Este grupo de critérios está dividido em quatro subclasses: **Especificação**, **Suporte a plataformas** robóticas, **Infraestrutura** e **Implementação**.

A subcategoria relacionada **Especificação** trata da fase de projeto de um sistema robótico, com ênfase na arquitetura de software. Os critérios de *neutralidade* de arquitetura e *primitivas arquiteturais* tratam da capacidade de utilizar diferentes arquiteturas de software e da existência de componentes pré-definidos como métodos de controle, motor de comportamento, máquinas de estado e etc. Como o sistema de visão seria apenas um destes componentes, o uso destes critérios não teria efeito no desenvolvimento. Por outro lado, o critério de *engenharia de software* tem importância. Principalmente porque o uso de metodologias de desenvolvimento de software e padrões de projeto de software contribui para para que códigos desenvolvidos por outros pesquisadores possam ser reutilizados.

Os critérios da subclasse de **Suporte a plataformas** advêm principalmente da necessidade de atender as diversas configurações de sensores e atuadores que os robôs podem ter. No desenvolvimento do sistema de visão não há uma definição sobre o uso de uma plataforma robótica específica, de forma que o critério de *suporte de hardware* não é necessário. O critério de *sistema operacional* (SO) também foi descartado uma vez que este não é preponderante no desenvolvimento. Eventualmente o não funcionamento de uma biblioteca de software desejada em um determinado sistema operacional poderia ser determinante. Porém, neste trabalho, apenas a biblioteca *OpenCV* foi utilizada na implementação do sistema de visão, e esta é compatível com os principais SOs.

Outro critério desconsiderado foi o de *método de configuração*. A configuração do robô durante o desenvolvimento do sistema de visão não é alterada constantemente. Uma eventual recompilação no caso de um ADR que exija alteração no código fonte seria um empecilho, porém não preponderante na escolha da ferramenta. No contexto original da metodologia, o critério de *simulação* buscava indicar a capacidade de emular sensores e atuadores das mais diversas plataformas de hardware. Considerando o foco deste trabalho em visão computacional, é indispensável a capacidade de construir

um ambiente tridimensional com alta fidelidade de modo a ser possível simular câmeras. Esta importância tornou este critério em um **pré-requisito** para a avaliação de um ADR.

As funcionalidades de um ADR relacionadas à comunicação e controle de execução são agrupadas na subclasse de **Infraestrutura**. Kramer e Scheutz (2007) incluíram nesta subcategoria critérios ligados à computação distribuída, considerando os *mecanismos* de distribuição, *mobilidade* dos componentes e *segurança*, além do critério de *escalabilidade*. Considerando que o objetivo atual se concentra em robôs individuais, estes aspectos têm, no momento, importância limitada. Por outro lado, ferramentas para controle e verificação de execução que provejam sistemas de registro (log), e depuração (debugging) são importantes para acompanhar a evolução de um código durante sua operação. Por isso, foram agregados no critério de *registro/depuração*. Outro critério relevante é a *comunicação de baixo nível* utilizada pelos ADRs na comunicação entre processos. Como serão utilizadas imagens, há interesse que o tráfego destas seja o mais rápido e simples possível.

A última subclasse de critérios funcionais está relacionada à **Implementação**. Na prática, a escolha de um ou outro ADR tem como objetivo facilitar o desenvolvimento da aplicação robótica. Nesta subcategoria destacam-se os critérios de *linguagem de programação* e *documentação*. O primeiro critério é importante no sentido de permitir ao pesquisador utilizar a linguagem que este tem preferência, facilitando assim a realização das implementações. O segundo critério aborda a existência de manuais de usuário e documentação do código-fonte e da API, os quais são de extrema importância pois auxiliam no uso de todas as funcionalidades do ADR.

Os critérios ignorados nesta subcategoria estão novamente no escopo do desenvolvimento de um sistema robótico completo. Destes destacam-se os critérios de *componentes predefinidos* e *linguagem de alto-nível*. O critério de operação em *tempo-real* foi descartado, pois não houve nesta etapa de pesquisa preocupações com aspectos temporais do sistema de visão para robôs, os quais poderão ser objetivo de trabalhos futuros. A exigência de um simulador tridimensional induz a existência de uma interface gráfica para mostrar o ambiente e as imagens das câmeras. O uso desta para controle durante a execução de testes pode ser benéfico, porém também representa mais uma fonte de erros de programação, sendo dispensáveis nesta forma. Como consequência destes fatos, o critério de *interface gráfica* tornou-se desnecessário. O último critério desta subclasse está relacionado a *integração de software*, que é suficientemente coberto pelo critério de comunicação de baixo nível, da subcategoria de infraestrutura.

b) Critérios de Usabilidade

Tão importantes quanto os critérios funcionais, os critérios desta classe representam fator decisivo na adoção de um ADR. Eles estão atrelados principalmente à facilidade de instalação e execução de experimentos. A avaliação dos critérios desta classe está profundamente sujeita à percepção do usuário durante o uso das ferramentas. Por isso todas as subclasses têm um critério para avaliar a experiência em cada um dos casos. A especialização no desenvolvimento apenas de um componente de visão computacional novamente exigiu modificações nos critérios originais da metodologia. A necessidade de um ambiente tridimensional de alta fidelidade associada a simplicidade da arquitetura requerida para os testes foram os principais motivos. Em consequência disto, abandonou-se a avaliação da subcategoria de **Usabilidade avançada**, que visa justamente a implementação de sistemas mais complexos. Deste modo, foram utilizadas apenas as subclasses de **Instalação** e **Usabilidade Básica**.

Dificuldades na **Instalação** de um ADR tem um efeito desencorajante em usuários potenciais, por isso é importante avaliar a maneira como esta ocorre através de uma subclasse específica. A *documentação* do processo de instalação é o critério mais importante nesta classe. Independente da complexidade deste processo, se este estiver bem descrito com instruções passo-a-passo, então a instalação torna-se facilitada. Também é desejável que existam listas de e-mails para que soluções de problemas encontrados por outros usuários possam ser compartilhadas.

Geralmente os ADRs requerem a instalação de componentes adicionais, os quais muitas vezes são comumente encontrados em computadores utilizados para desenvolvimento de software. Este é o caso de compiladores e bibliotecas de desenvolvimento de uso amplo. Porém, em muitos casos há a necessidade de instalar componentes menos conhecidos, o que exigem um trabalho maior. Por isso, é necessário utilizar o critério de *instalação de componentes externos* ao ADR para indicar a existência desta complexidade. Obviamente a *instalação do próprio ADR* também é um critério, principalmente no caso de compilação do código-fonte onde é possível encontrar "bugs" e incompatibilidades com versões de bibliotecas. Por último nesta subclasse, há o critério que qualifica a *experiência* durante o processo de instalação. Como já explicado, este é um critério um tanto quanto subjetivo, sendo neste caso, dependente dos conhecimentos do usuário, bem como da versão específica que está sendo testada. As vezes, uma nova versão do ADR tem atualizações que resolvem problemas antigos, e em outros casos, a adição de novos recursos acrescenta novas dificuldades na instalação.

A subcategoria de **Usabilidade Básica** tem como objetivo avaliar a implementação e execução de arquitetura robótica simples. Na proposta original esta arquitetura consistia de um robô com movimento aleatório (*wander*

behaviour) com capacidade de desviar obstáculos. O objetivo ao usar esta arquitetura era testar o uso de um "mínimo denominador comum" aos ADR. Este mínimo foi definido como a capacidade de movimentar o robô e acessar os dados de sensores, no caso um *rangefinder*. Tendo em vista o foco em visão computacional, alterou-se o teste, o qual passou a objetivar a extração de imagens a partir de uma ou mais câmeras acopladas ao robô e controle manual de posição do mesmo. Desta maneira o foco em simulação fez com que a capacidade de implantação do código criado em um robô não fosse avaliada.

Concentrou-se os critérios de *documentação* e *implementação* no último, pois o desenvolvimento da arquitetura é feita principalmente através do uso de tutoriais e códigos de exemplo. O critério de *experiência* também reuniu os critérios de *overhead*, *ferramentas gráficas* e *execução* da metodologia original. Isso ocorreu devido à avaliação mais subjetiva, feita durante a utilização do ADR, na qual foi observada a maneira como os testes ocorreram, os problemas (overhead) e as facilidades encontradas (ferramentas gráficas).

Para avaliar a capacidade de simulação de ambientes tridimensionais foram adicionados dois critérios. O primeiro avalia o ADR quanto as ferramentas para *criação do ambiente* de simulação. O segundo tem como objetivo avaliar a *qualidade gráfica* do ambiente criado. Neste último considera-se as capacidades que o ADR disponibiliza ao ser instalado, isto é, sem que seja necessário fazer modificações/adições no código-fonte do ADR.

c) Critérios de Influência

A influência dos ADRs na comunidade científica também é significativa na avaliação dos ADRs. Porém, esta classe tem um viés mais indicativo das capacidades da ferramenta. Neste contexto, o critério de *utilização* busca avaliar o uso do ADR em projetos de pesquisa. Os autores da metodologia original fizeram esta análise observando em quais sub-áreas da robótica cada ADR avaliado teve influência. O uso destes em apenas uma publicação em uma determinada sub-área foi considerada suficiente para indicar o ADR como impactante na mesma. Este critério foi relaxado na presente avaliação, onde a simples existência de trabalhos usando o ADR já foi considerada suficiente. Por último, o critério de *compatibilidade* tem como objetivo avaliar a capacidade de relacionamento do ADR com outras ferramentas.

3.1.2 Avaliação

Ao utilizar esta metodologia para avaliar nove ADRs, Kramer e Scheutz (2007) tinham como objetivo identificar os pontos fortes e fracos de cada ferramenta apenas para compará-las, sem a intenção de indicar ou escolher

uma como a melhor. Eles deixam claro que os objetivos do pesquisador interessado em usar este tipo de ferramenta são fatores cruciais na escolha de uma ou outra opção de ADR. Para qualificação quantitativa, eles sugerem geralmente o uso de valores ternários, indicando a **não-conformidade** ou a **conformidade parcial/total** com determinado critério. Desta forma, a partir do pré-requisito de *simulação* tridimensional dois ADRs foram selecionados para avaliação:

- MORSE[1]: Desenvolvido na França, no *Laboratoire d'Analyse et d'Architecture des Systèmes* (LAAS) no contexto do projeto Openrobots. Baseado no Blender Game Engine;
- Gazebo[2] : Simulador desenvolvido pela *Open Source Robotics Foundation*(OSRF), principalmente para aplicações *outdoor*. Baseado no motor de renderização OGRE e motor físico Bullet;

Além de atenderem o pré-requisito, o fato de ambos possuírem suporte de entidades reconhecidas e de serem de código-aberto, contribuíram na escolha destas opções para serem avaliadas. Outro fator importante é a continua atualização de ambas as opções, as quais têm tido grande desenvolvimento recentemente. O MORSE teve quatro novas versões lançadas nos últimos dois anos. Porém a versão 1.0 foi lançada apenas no inicio de 2013. O Gazebo é significativamente mais antigo, tendo seu projeto iniciado em 2002. Esteve por muito tempo ligado ao projeto Player/Stage, sendo o simulador tridimensional deste. Teve um grande desenvolvimento nos últimos dois anos, sendo que apenas no último, oito novas versões foram lançadas. A versão 1.0.0 foi lançada na primeira metade de 2012.

As avaliações foram desenvolvidas entre fevereiro e março de 2012, utilizando as versões mais atuais à época. No caso do Gazebo foi testada a versão 1.0.0-RC2, lançada em 30 de janeiro de 2012. Já no caso do MORSE foi usada a versão 0.5.1, disponível a partir de 1^o de março de 2012. A sequência na qual foram analisados os critérios e os resultados desta avaliação estão descritos na tabela 5.

a) Influência

Iniciou-se a avaliação pelos critérios de influência, porque parte das informações relativas a estes já estavam disponíveis quando se fez a busca pelos ADRs. Quanto ao critério de **utilização**, Echeverria et al. (2011) cita o uso do MORSE em três projetos de pesquisa. Destaca-se o projeto *ACTION*, que foca na cooperação entre robôs de tipos heterogêneos na tarefa de localização

[1] `http://www.openrobots.org/wiki/morse/`
[2] `http://gazebosim.org/`

Tabela 5 – Avaliação dos ADRs Gazebo e MORSE.

Subclasse	**Critérios**	**Gazebo**	**MORSE**
Influência	Utilização	⊞	⊟
	Compatibilidade	⊟	⊞
Instalação	Documentação	⊟	⊟
	Componentes externos	⊟	⊡
	Próprio ADR	⊞	⊞
	Experiência	⊟	⊟
Implementação	Linguagem de Programação	⊞	⊟
	Documentação	⊞	⊞
Especificação	Engenharia de Software	⊞	⊞
Infraestrutura	Registro/Depuração	⊞	⊞
	Comunicação	⊞	⊟
Usabilidade Básica	Implementação de teste	⊟	⊞
	Criação do ambiente	⊡	⊞
	Qualidade gráfica	⊡	⊡
	Experiência	⊡	⊞

Símbolos: ⊡ : não conforme ⊟ : parcialmente conforme ⊞ : conforme

e detecção em ambientes externos complexos, considerando vários cenários. Porém, não foram encontradas publicações demonstrando a aplicação deste ADR em tais projetos. Por isso, o MORSE foi considerado parcialmente conforme no critério de utilização.

No caso do Gazebo, destaca-se o uso deste como base para o simulador usado no *DARPA Robotics Challenge*. O principal objetivo desta competição é o "desenvolvimento de robôs capazes de desempenhar tarefas complexas em ambientes perigosos, degradados ou construídos por humanos".[3] Além disso, destaca-se o uso deste ADR em publicações recentes, como na simulação de um veículo aéreo não tripulado (MEYER et al., 2012). Desta forma, foi atribuído ao Gazebo valoração completa no critério de utilização.

Para avaliar a **compatibilidade** dos ADRs, decidiu-se usar o número de *middlewares* com os quais estes são compatíveis. Foi escolhida esta mé-

[3] `http://www.darpa.mil/Our_Work/TTO/Programs/DARPA_Robotics_Challenge.aspx`

trica devido à dificuldade de verificar todos o projetos que têm compatibilidade com cada ADR. Como a informação relativa aos *middlewares* é facilmente acessível, esta foi escolhida como indicador. O MORSE suporta nativamente quatro *middlewares* (ROS, YARP, MOOS e Pocolibs), enquanto o Gazebo apenas dois (ROS e Player). Assim, o MORSE recebeu o valor máximo, enquanto o Gazebo apenas parcial.

b) Instalação

Os parâmetros da subclasse de instalação foram avaliados na sequência, pois uma possível incapacidade neste processo tornariam irrelevantes qualquer outro critério. Tanto o MORSE quanto o Gazebo são ferramentas desenvolvidas para o sistema operacional Linux, de modo que se tentou instalá-los apenas neste SO. Para referência, foram usadas três distribuições: Archlinux, Fedora 16 e Ubuntu 11.10, todas em sua última versão a época.

Quanto ao critério de **documentação**, o Gazebo tem um foco maior no Ubuntu, e por isso a documentação apresentou o nome das dependências de acordo com a nomenclatura usada por esta distribuição. Isto tornou instalação problemática na outras distribuições. Observou-se que pequenos detalhes quanto à ordem de algumas configurações estavam incorretas, o que causou algumas complicações, e consequentemente, o Gazebo obteve uma conformidade parcial no critério de documentação de instalação. O MORSE, por ser mais independente de distribuição, permitiu que fosse possível instalá-lo em todas as distribuições. O único problema quanto à documentação foi a omissão de uma etapa de pré-instalação relacionada a um *script* propietário, o que se provou difícil de ser descoberto. Por isso, o MORSE também recebeu uma qualificação parcial neste critério.

Quanto à instalação de **componentes externos**, o Gazebo não apresentou problemas no Ubuntu. No Fedora, o maior problema foi encontrar o nome correto para as dependências necessárias, como já explicado na avaliação da documentação. Houve problemas no Archlinux ao instalar o motor de renderização OGRE, o qual já estava em uma versão mais recente do que a suportada pelo Gazebo. Assim, este ADR foi considerado parcialmente conforme na instalação de componentes externos. Com relação ao MORSE houve problemas com a instalação do Blender. Apenas o Fedora tinha versão compatível no repositório, enquanto o Archlinux tinha uma versão mais recente e o Ubuntu uma versão mais antiga. Por isso, nestas duas distribuições foi necessário instalar o Blender a partir dos arquivos fornecidos diretamente por seus desenvolvedores.

Outro componente externo necessário para executar o testes de usabilidade foi o YARP, um *middleware* de robótica. O Fedora possui o método mais prático para instalação deste componente, pois não há necessidade de

compilar. Tanto o Archlinux quanto o Ubuntu exigiram compilação deste componente, sendo que o primeiro ainda exigiu a compilação de outra biblioteca. Considerando a maior complexidade ao instalar os seus componentes externos, o MORSE foi julgado não conforme neste critério. Levando em conta as considerações sobre a documentação e instalação de componentes externos, o processo de **instalação próprio** de ambos os ADRs não teve problemas, o que levou a uma conformidade total neste critério.

Como um todo, a **experiência** de instalar ambos os ADR foi considerada regular. Esperava-se um certo grau de dificuldade quanto as versões dos componentes necessários, o que efetivamente ocorreu. Além disso, não houve problemas insolúveis de compilação que poderiam impedir a avaliação. Baseado nestas afirmações e nos critérios de instalação anteriores, ambos os ADRs receberam o valor intermediário quanto a experiência. Além disso, ponderando as dificuldades encontradas durante a instalação, decidiu-se utilizar o Ubuntu como referência para o restante da avaliação, por ser a distribuição mais conhecida.

c) Implementação

Seguindo para a subclasse de implementação, foi avaliado inicialmente o critério de **linguagem de programação**. Devido ao maior conhecimento em C++, há uma preferência pelo desenvolvimento nesta linguagem. Outro fator, é o interesse em usar a biblioteca de visão computacional *openCV*, a qual têm um suporte mais completo a esta linguagem. Por isso, definiu-se o suporte a C++ como o principal fator na avaliação deste parâmetro. O Gazebo é programado nativamente em C++, sendo assim considerado totalmente conforme. O MORSE é feito em *python*, sendo necessário o uso de *wrappers* para utilizar o C++. Com isso, este ADR recebeu parecer parcialmente conforme neste quesito. Já em relação ao critério de **documentação**, ambos os ADRs receberam valoração máxima. Ambos apresentam uma extensa documentação, abrangendo desde as APIs de código e manuais, até sugestões e *how-to*s.

d) Especificação

Avançando para a subclasse de especificação, verificou-se em ambos os ADRs uma grande preocupação com aspectos de **engenharia de software**. Tanto o MORSE quanto o Gazebo tem publicações dedicadas à apresentação dos mesmos (Echeverria et al. (2011) para o MORSE e Koenig e Howard (2004) para o Gazebo), que tratam sobre a descrição da arquitetura proposta, apresentando a estrutura de classes, diagramas de fluxo de dados e etc. Em função disto, e de toda a documentação já discutida na subclasse de implementação, os dois ADRs receberam grau de conformidade total neste critério.

e) Infraestrutura

Ao avaliar as subclasses de instalação e implementação, também se obteve as informações relativas a subclasse de infraestrutura. Com relação as ferramentas de **registro e depuração**, ambos os ADRs obtiveram grau de conformidade total. No MORSE pode-se utilizar as ferramentas disponibilizadas através da própria linguagem python, o que se julgou suficiente. Como o Gazebo é feito em C++, a depuração é feita naturalmente através do *gdb* (GNU Debugger). Para melhor funcionamento é necessário compilar este ADR utilizando a opção de compilação para depuração. Quanto ao registro, o Gazebo oferece ferramentas para registro através das funções `gzlog` e `gzmsg`.

Boa parte do critério de **comunicação** já foi avaliado, ao verificar a compatibilidade da subclasse de influência. Para comunicações simples, isso é, de mensagens contendo textos, o MORSE permite o uso dos quatro *middlewares* já comentados, além da utilização de *sockets* e arquivos de texto. Para transferência de imagens, apenas dois destes métodos de comunicação não podem ser utilizados: MOOS e arquivos de texto. O Gazebo oferece menos middlewares de comunicação, porém oferece a possibilidade de utilizar `plugins`, os quais permitem o controle e interação diretas com todos os aspectos da simulação. Este atributo do Gazebo fornece grande flexibilidade na utilização e transferência de imagens, garantindo a valoração máxima deste ADR neste quesito. Comparativamente o MORSE não apresenta característica parecida e portanto recebeu avaliação apenas parcial.

f) Usabilidade

A subclasse de usabilidade foi a última a ser analisada. Usando os tutoriais disponibilizados pelos dois ADRs, foi feita a **implementação do teste** de uso descrito na seção 3.1.1. O MORSE recebeu melhor avaliação porque foi possível desenvolver o teste utilizando apenas os tutoriais. Estes cobriram a obtenção de imagens através do *middleware* YARP e um atuador do tipo *waypoint*, que recebe como referência a posição final desejada. Novas referências podem ser informadas durante a execução, garantindo o controle manual online de posição. Os tutoriais do Gazebo permitiram apenas a obtenção de imagens através de uma trajetória pré-definida no código-fonte do *plugin* implementado para o teste. Como não foi possível fazer o controle *online* de posição apenas com os tutoriais, este ADR recebeu valor parcial para este critério.

Do ponto de vista das capacidades dos dois ADRs, a **criação do ambiente** de teste foi onde se percebeu a maior disparidade. Como o MORSE é desenvolvido sobre o Blender, ele se aproveita deste para prover uma interface completa de modelagem tridimensional. Devido ao elevado número

de ferramentas disponíveis, esta interface se torna significativamente complicada para usuários leigos em computação gráfica. Mesmo assim, foi atribuído ao MORSE valor máximo nesta categoria.

Já o Gazebo apresentou ferramentas muito simples para a construção de ambientes. Na versão testada, a construção de objetos era bastante limitada, sendo feita apenas através da composição de formas simples como cuboides, cilindros ou esferas. Além disso, para se utilizar objetos baseados em malhas poligonais anteriormente existentes, é necessário utilizar um arquivo de descrição compatível com o formato SDF (*Simulation Description Format*), usado pelo Gazebo. Isto dificulta o uso de um software de modelagem tridimensional para construir o ambiente. Por estas razões o Gazebo recebeu grau de não conformidade quanto a construção do ambiente.

Ao realizar os testes, observou-se que a **qualidade gráfica** dos ambientes-exemplo disponíveis, ou aqueles criados usando os objetos disponibilizados pelos ADRs, foi inferior ao desejado. Destaca-se aqui a importância de avaliar a aparência das sombras. A ausência de luz causada pela interferência de objetos na trajetória da luz, é uma importante fonte de ruídos para os sistemas de visão computacional. Por isso é interessante que os simuladores dos ADRs consigam implementá-las da melhor maneira possível.

No caso do MORSE, é possível perceber o excessivo serrilhamento das sombras na figura 9. Na figura 10, referente ao Gazebo, é possível ver o a ausência de *self-shadowing*, isto é, o efeito de um objeto projetar sombras sobre si próprio. Outro problema no Gazebo, foi a rigidez das sombras, as quais não apresentam o esmaecimento característico da penumbra de uma sombra. Além disso, a qualidade das texturas utilizadas em ambos os casos deixa à desejar. Considerando estes fatos, foi atribuído tanto ao Gazebo quanto o MORSE o valor mínimo neste critério.

Finalizando, houve considerável disparidade no quesito de **experiência** de usabilidade. Os únicos pontos negativos no caso do MORSE foram o aparecimento do efeito *gimball lock* algumas vezes ao mudar a pose da visão livre, e o excessivo numero de processos necessários para executar os testes. Durante o teste de captura de imagens, até quatro terminais foram necessários para controle e observação da simulação. Devido ao pouco impacto que estes problemas causaram, o MORSE recebeu o grau máximo neste critério. Por outro lado, no Gazebo houve problemas significativos. Além do constante acontecimento de falhas de segmentação de memória e exceções de execução, houve problemas no posicionamento de objetos. Ás vezes, ao se inserir um objeto, este aparecia apenas na árvore de modelos, sem aparecer no ambiente. Por vezes, ao deletar estes objetos, causava-se falhas de execução já relatadas. Além disso, alguns objetos se moviam de maneira indesejada para outros locais, após serem posicionados. Considerando a gravidade destes pro-

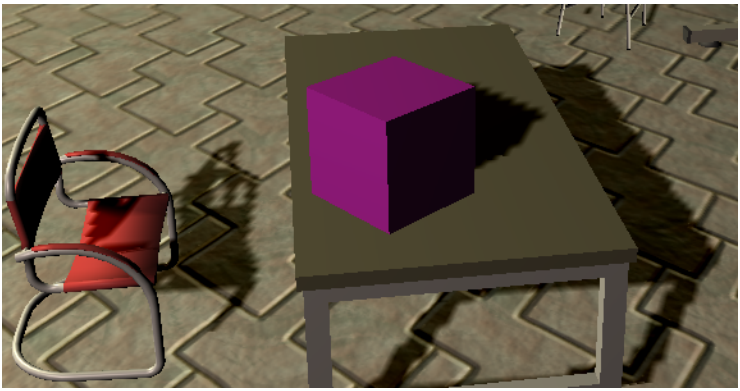


Figura 9 – Exemplo da qualidade gráfica no MORSE. Note o serrilhamento das sombras e a simplicidade das texturas utilizadas no ambiente-exemplo.

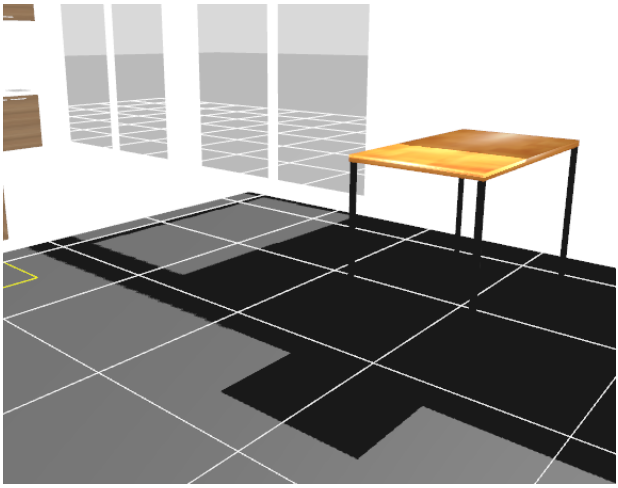


Figura 10 – Exemplo da qualidade gráfica no Gazebo. Note a falta de continuidade da sombra no piso branco. Isso ocorre pois o algoritmo de geração de sombras não consegue tratar a situação onde uma malha poligonal deve projetar sombras sobre si própria. Neste caso a parede e o piso fazem parte da mesma malha. Além disso, percebe-se a rigidez da sombra sobre a mesa.

blemas, não foi possível atribuir valor maior que o mínimo para o Gazebo no critério de experiência.

Observando os resultados da avaliação, é possível verificar resultados distintos. Nota-se que os critérios de implementação e infraestrutura apontam o Gazebo como o ADR mais indicado. Isso ocorre principalmente por este usar a linguagem de programação desejada (C++), e fornecer através dos *plugins*, a maneira mais eficiente e flexível para extração de imagens. Já na subclasse de usabilidade, é nítida a vantagem do MORSE, o qual recebeu valor máximo em quase todos os critérios. Isto se deve principalmente a melhor interface, consequência da construção deste ADR sobre o Blender. Por este ser por si só um software de modelagem tridimensional, ele é capaz de transferir ao MORSE melhores ferramentas para modelagem dos ambientes. Porém, em função dos problemas de qualidade gráfica, ficou claro que não seria possível usar nenhum dos dois ADRs avaliados sem que maiores estudos na área de computação gráfica fossem realizados.

3.2 DESENVOLVIMENTO DO SIMULADOR

A análise efetuada permitiu obter noções sobre os motores gráficos usados por ambos os ADRs. Principalmente com relação ao OGRE, que é o motor gráfico usado pelo Gazebo. As informações sobre este são de fácil acesso, sobretudo através do manual, mas também por meio da comunidade de usuários (fórum e *wiki*). Inclusive, há grande quantidade de código disponível para uso geral da comunidade. A partir das informações obtidas nestas fontes, descobriu-se soluções para os problemas gráficos relatados anteriormente. Ficou evidente a necessidade de implementar estas soluções ou, integrar implementações construídas por outros desenvolvedores, para tornar útil, do ponto de vista de qualidade gráfica, qualquer dos ADRs avaliados anteriormente.

Rapidamente se percebeu que seria significativamente complicado aplicar as soluções encontradas diretamente em qualquer um dos ADRs. Principalmente devido ao tempo necessário para compreender o código-fonte de cada ADR. Esta situação seria exacerbada no caso do MORSE, pois seria necessário ainda compreender a maneira como Blender renderiza as sombras e texturas. Este tem uma documentação extremamente abrangente com relação as ferramentas disponibilizadas, porém, significativamente menos acessível quanto ao funcionamento do motor gráfico do BGE (*Blender Game Engine*). Estudando os tutoriais do OGRE, notou-se uma relativa facilidade para criar as aplicações com funcionalidades suficientes para executar a simulação de robôs em ambientes tridimensionais. Assim, decidiu-se desenvolver um si-

mulador utilizando as características mais vantajosas de cada um dos ADRs analisados na seção 3.1.2:

- Do MORSE derivou-se o uso do Blender para modelagem do ambiente de simulação. Este possui melhores ferramentas que o Gazebo para modelagem tridimensional.
- Do Gazebo derivou-se o uso do motor gráfico OGRE para renderização. O principal motivo é a grande disponibilidade de documentação, tutoriais e exemplos de códigos.

Assim, partiu-se para definição do fluxo básico de operações necessárias para disponibilizar imagens simuladas para execução de testes. O fluxograma da figura 11 demonstra a sequência de operações necessárias para atingir este objetivo. A primeira tarefa é a busca por malhas poligonais dos objetos que serão usados para construção do ambiente. Levando em conta os objetivos relacionados à robótica, decidiu-se pela simulação de um ambiente com a aparência de um escritório. Isso porque este tipo de ambiente oferece os desafios esperados quanto a localização de objetos e definição de trajetórias. A partir da escolha do tipo de ambiente a ser simulado, buscou-se as malhas poligonais principalmente em base de dados do campo da arquitetura[4]. Após a seleção das malhas poligonais, a segunda tarefa foi construir o modelo do ambiente utilizado o Blender.

A tarefa de construção do ambiente é constituída principalmente da criação das malhas poligonais adicionais e configuração de todas as malhas que formam o ambiente. As malhas poligonais adicionais são aquelas que definem as paredes, o chão e o teto do ambiente, as quais têm formas bastante simples, porém, são específicas demais para serem encontradas na internet. A configuração das malhas consiste na definição dos parâmetros da mesma, como, por exemplo, a posição, a escala e orientação destas. Além disso é necessário definir o(s) material(s) de cada malha e os mapas para uso de textura. Terminada a construção do modelo do ambiente, este precisa ser exportado para um formato compreensível pelo OGRE. Para isso utilizou-se a ferramenta *blender2ogre*[5], que converte a malhas para o formato usado pelo OGRE e cria um arquivo `*.scene`, escrito em *XML*, que descreve o ambiente criado.

Já na parte relativa aos processos internos ao programa de simulação, a tarefa seguinte é a montagem do ambiente no contexto do renderizador. Para isto é utilizada a classe `DotSceneLoader`, disponibilizada pela comunidade do OGRE. Esta classe faz a análise sintática do arquivo `*.scene` que descreve o ambiente, e instancia os objetos (referências) necessários para que as

[4] `http://www.archibase.net/`

[5] `https://code.google.com/p/blender2ogre/`

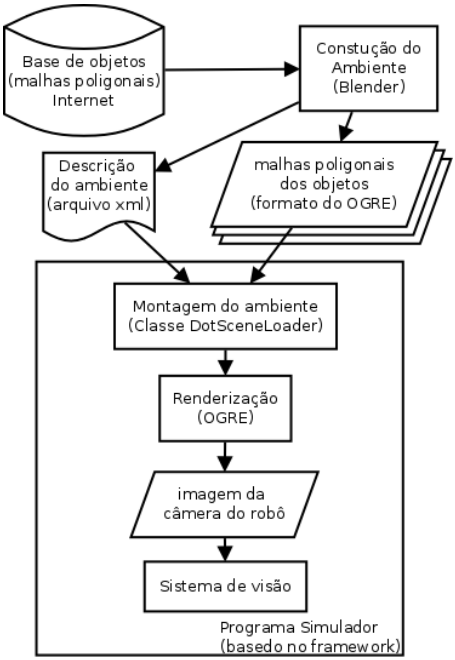


Figura 11 – Fluxograma de uso do simulador

malhas poligonais apareçam corretamente na renderização. Considerando a implementação do programa de simulação, a última tarefa deste é gerar as imagens observadas à partir das câmeras do robô. Esta tarefa representa a fonte de imagens para a aplicação de visão computacional embutida no robô, e exige a consideração de aspectos de concorrência entre *threads* no simulador e a conversão da imagem para um formato utilizável pelas biblioteca de visão computacional.

A seguir serão apresentadas informações mais aprofundadas sobre o implementação do simulador. Inicialmente será apresentado o *framework* desenvolvido para ser a base para implementação de simuladores para aplicações de visão computacional. Este foi baseado no *framework* disponibilizado pelos desenvolvedores do OGRE para implementação de seus tutoriais. Serão detalhadas a estrutura de classes do *framework* e as principais funções da sequência de operações de controle de simulação, que ocorrem durante o laço de renderização. Posteriormente serão abordados as soluções encontradas para os problemas de qualidade gráfica.

3.2.1 Framework para o desenvolvimento de simulações

A maior parte do conhecimento adquirido em relação ao motor gráfico OGRE, foi obtido por meio da implementação dos tutoriais sugeridos na documentação deste. A implementação dos tutoriais forneceu peças de código com funcionalidades suficientes para permitir o desenvolvimento de uma grande variedade de aplicações. Construindo abstrações sobre as classes do próprio OGRE ou do código dos tutoriais foi possível elaborar um *framework* para o desenvolvimento de simulações para aplicações de visão computacional. Um exemplo das abstrações criadas é a classe `SimCamera`, a qual encapsula a classe `Camera` do OGRE. A nova abstração adiciona um método para captura as imagens em formato compatível com a biblioteca de visão computacional utilizada no desenvolvimento das tarefas. Este método tem a particularidade de considerar a possibilidade de concorrência entre a *thread* que fez a requisição da imagem e a *thread* de renderização.

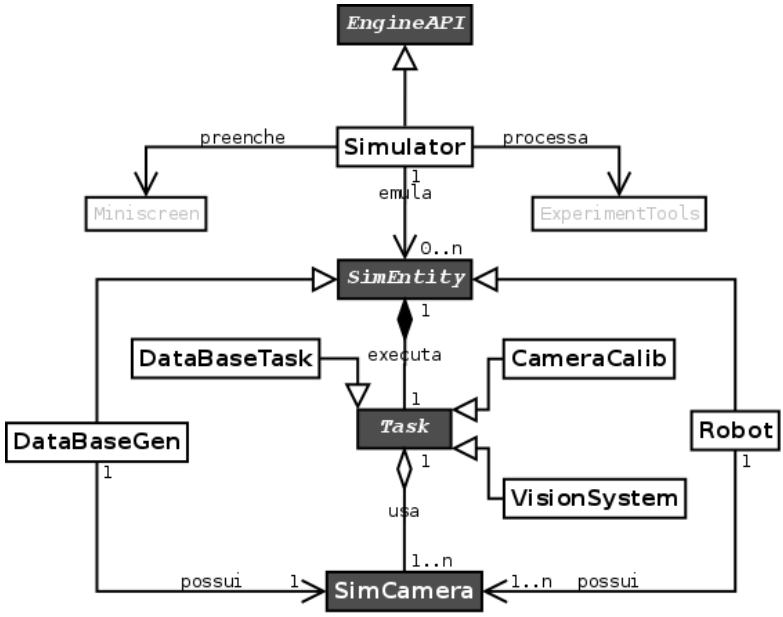


Figura 12 – Diagrama de classes da aplicação. As classes do *framework* estão em fundo escuro. Classes abstratas estão com fonte em itálico. Classes auxiliares com fonte em cinza.

O objetivo do *framework* é minimizar o número de classes que precisam ser criadas para elaborar um simulador tridimensional para visão. Deste modo, chegou-se a um formato onde é necessário criar o mínimo de três classes para obter um simulador: (i) uma classe para montagem do ambiente e instanciação de entidades; (ii) uma classe para representar entidades que realizam ações no ambiente; e (iii) uma classes para implementar estas ações. As classes que se encaixam em alguma destas descrições serão classificadas como **classes-objetivo**. A figura 12 mostra o diagrama de classes com o relacionamento entre as classes do *framework*. Também estão nesta figura as classes-objetivo elaboradas sobre o *framework*.

Para compreender o *framework* é necessário começar pela classe abstrata `EngineAPI`, que é derivada da classe `BaseApplication` presente nos tutoriais. Neles, esta classe demonstra uma série de interfaces para os métodos disponibilizados pelo renderizador. Foram feitas modificações no sentido de diminuir a quantidade de métodos, retirar e acrescentar propriedades desejadas. Através de herança, criou-se a classe `Simulator`, que representa o primeiro tipo de classe-objetivo. Como qualquer classe-filha, esta classe precisa sobrecarregar os métodos abstratos da classe `EngineAPI`. Entre estes métodos destaca-se o método para montagem do ambiente tridimensional. Além disso, esta classe possui métodos para modificação das propriedades das entidades passivas. Como descrito anteriormente, este tipo de classe também é responsável pelos métodos para instanciação das entidades ativas do simulador.

As entidades ativas são aquelas do segundo tipo de classe-objetivo. No contexto do *framework* elas são representadas pela classe abstrata `SimEntity`. Esta abstração foi construída sobre a classe `Entity` do OGRE. A classe `Entity` armazena a malha poligonal de um objeto do ambiente, representando o "corpo" deste objeto. O principal objetivo da classe `SimEntity` é permitir que as entidades ativas possam tratar eventos gerados pelo renderizador ou pelo teclado. Outro ponto importante da classe `SimEntity` é que esta tem tarefas em composição. As tarefas, definidas pela classe abstrata `Task`, são a terceira classe-objetivo do *framework*, e representam a única abstração original do mesmo.

Como não se tem pretensões quanto a execução de tarefas diferentes simultaneamente, e visando a simplicidade, cada entidade só pode executar uma tarefa de cada vez. Devido ao objetivo de implementar tarefas relacionadas a visão computacional, todas as tarefas exigem o uso de câmeras. Por isso há a agregação entre as classes. Nota-se que as câmeras não pertencem as tarefas, e sim as entidades. A classe `Task` implementa vários métodos, inclusive um para captura de imagens de múltiplas instâncias da classe `SimCamera`. Além desse método, ela também fornece uma função para ex-

tração das imagens com os resultados das tarefas. A classe `Simulator` usa este último método para atualizar os quadros de apresentação dos resultados, implementados pela classe `Miniscreen`.

É importante salientar que as classes `DataBaseTask`, `CameraCalib` e `VisionSystem`, filhas de `Task`, executam suas tarefas por meio de *threads* próprias, no mesmo processo de sistema do simulador. Esta é uma grande diferença com relação aos ADRs avaliados anteriormente, principalmente no caso do MORSE, pois estes faziam uso *middlewares*, *sockets* e etc para compartilhamento de dados. A concorrência entre as *threads* de renderização e execução de tarefas, tornou necessário garantir a exclusão mútua durante a leitura/escrita de imagens compartilhadas. Isso inclui as imagens obtidas através da classe `SimCamera`, além das imagens com o resultados das tarefas. Por fim, para executar automaticamente os testes, o simulador faz uso da classe `ExperimentTools`, que faz a análise sintática de arquivos *XML* contendo os parâmetros dos testes.

O fluxo de execução das aplicações desenvolvidas sobre o *framework* foi pensado para privilegiar a velocidade de renderização. Assim, foram usados os eventos gerados pelo OGRE para controlar a sequência de operações necessárias para evolução da simulação. A classe `EngineAPI` forneceu uma interface a funções que são invocadas na ocorrência de diversos tipos de eventos relacionados aos vários processos de renderização e com os métodos de entrada.

Um evento importante é o que indica a atualização de um alvo de renderização. Um alvo de renderização é um espaço de memória que funciona como um "tela de pintura" para o renderizador. O OGRE os usa em várias funções. No contexto do *framework* eles são usados para implementar o buffer de saída das câmeras da classe `SimCamera`. Toda vez que o renderizador cria a imagem que representa a visão instantânea de uma câmera, ele a deposita no alvo de renderização da câmera correspondente e invoca o método `postRenderTargetUpdate`. Assim, toda vez que um alvo de renderização de uma `SimCamera` é atualizado, é preciso executar o método que converte a imagem do formato do OGRE, para o formato `Mat` utilizado pelas tarefas.

A figura 13 exemplifica o tratamento deste evento. Neste exemplo foi usado um simulador com apenas uma entidade ativa (`Robot`), a qual possui apenas uma câmera. Inicialmente o método `postRenderTargetUpdate`, que a classe `Simulator` sobrescreve, é invocado. Este invoca o método `postRenderTargetUpdate` da entidade ativa `Robot`, para que esta possa verificar se a própria está relacionada com o evento de atualização que disparou a sequência. Em caso negativo, a função apenas retorna. Em caso positivo, `Robot` invoca o método `postRenderTargetUpdate` de `SimCamera` para que seja feita a conversão do formato da imagem.

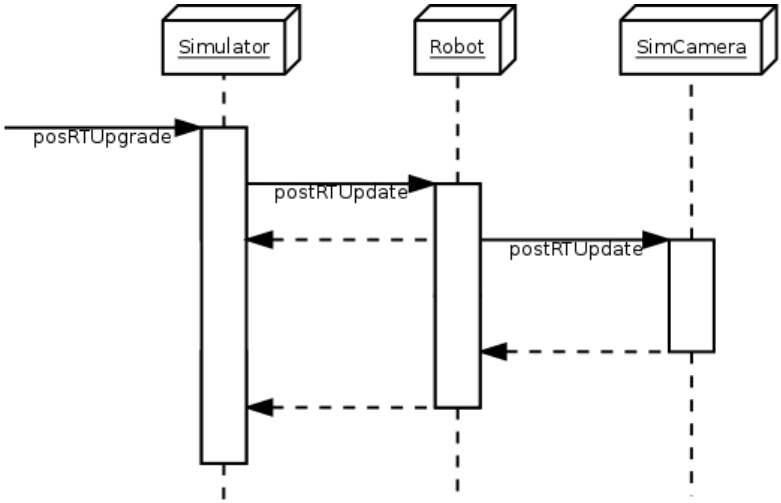


Figura 13 – Diagrama de sequência para o evento de atualização de alvo de renderização.

O principal evento utilizado para controle de execução ocorre quando os trabalhos de todos os alvos de renderização necessários para o quadro seguinte já foram enviados para a fila de processos do processador gráfico. Na ocorrência deste evento específico, a função `frameRenderingQueued` é invocada pelo renderizador. Há interesse neste evento porque após a ocorrência deste, o fluxo de renderização ficará bloqueado, esperando o processador gráfico terminar todas as renderizações requisitadas. Este fato faz com que a CPU fique ociosa durante este período. Por isso é interessante aproveitar o tempo para fazer o tratamento dos eventos dos periféricos de entrada, capturar a última imagem de saída, caso uma tarefa esteja em execução, e atualizar a posição dos objetos no ambiente.

Devido ao elevado número de operações relacionadas a este evento, será demonstrado apenas o caso de uso da tecla "W" para fazer um robô da classe `Robot` deslocar-se para frente, como é comum em aplicações de jogos. A figura 14 mostra o diagrama de sequência do tratamento deste caso.

Ao ser invocado pelo renderizador, o método `frameRenderingQueued`, implementado na classe `EngineAPI`, faz uso da função `capture`. Esta é disponibilizada pela biblioteca OIS (*Object Oriented Input System*), para captura de eventos de teclado. Se entre as invocações de `frameRenderingQueued`, alguma tecla tenha sido pressionada, então `capture` irá invocar `keyPressed`,

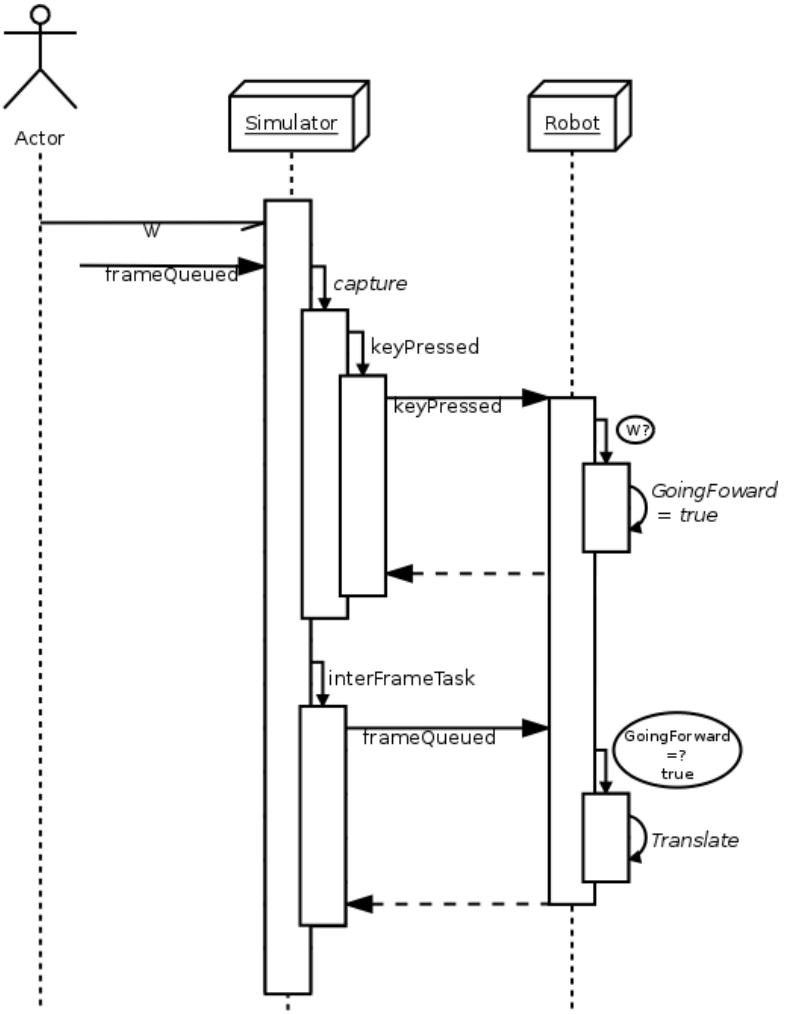


Figura 14 – Diagrama de sequência para deslocamento do robô atráves do teclado. Itens circulados representam tomada de decisão

um método sobrescrito por `Simulator`. Além de implementar o tratamento do pressionamento de teclas específicas, este método invoca `keyPressed` da classe `Robot`, a qual trata dos eventos de teclado relevantes a esta classe específica.

Caso "W" tenha sido pressionando, então `keyPressed` de `Robot` marcará o *flag* `mGoingForward` como verdade. Este *flag* é checado pelo método `frameRenderingQueued` do robô, quando este é invocado pela função `interFrameTasks` da classe `Simulator`. Esta por sua vez, é invocada pelo método `frameRenderingQueued` implementado na classe `EngineAPI`, já mencionado anteriormente. Se `mGoingForward` for verdadeiro quando checado pelo método do robô, então a posição do robô no ambiente será modificada de acordo com a cinemática definida para o robô. Atualmente o robô tem sua cinemática definida através de equações de movimento retilíneo uniforme.

Enquanto a tecla "W" não for solta, o robô continuará deslocando-se para frente. No momento que a tecla "W" é solta, a sequência de operações é idêntica. Porém, o método que a função `capture` irá invocar será `keyReleased`. Este método é análogo a `keyPressed`, só que tratando os eventos de soltura de teclas. Quando `keyReleased` do robô for invocado devido a soltura da tecla "W", o flag `mGoingForward` será marcado como falso, cessando o movimento do robô.

3.2.2 Aspectos gráficos

Uma vez descrita a criação do *framework* usado no desenvolvimento do simulador, ainda é necessário abordar os aspectos de computação gráfica envolvidos com a simulação. Dois problemas desta área foram encontrados durante a implementação do simulador: o **sombreamento** das malhas poligonais e a **geração de sombras**. É importante ressaltar que os estudos feitos nessa área não foram exaustivos. O objetivo foi atingir uma qualidade suficiente para a execução dos testes desejados, levando em conta as limitações de tempo de desenvolvimento e os conhecimentos/habilidades necessárias para desenvolver as soluções candidatas. Estes fatores foram determinantes para a decisão de criar o simulador através do renderizador OGRE.

Considerando a maior facilidade do acadêmico em programar em comparação a modelar malhas poligonais e textura, o uso do renderizador OGRE se mostrou mais adequado em relação ao Blender Game Engine, o qual tem maior foco em usuários da área de design gráfico. Como já salientado, a documentação disponibilizada pela comunidade de usuários do renderizador OGRE foi imprescindível para a obtenção dos resultados alcançado. Através do manual foi possível verificar quais eram as capacidades incluídas no renderizador, e como adicionar novas. O fórum de usuários foi uma fonte importante de trechos de código, os quais permitiram agilizar o desenvolvimento da solução final.

a) Sombreamento

O primeiro aspeto gráfico atacado durante o desenvolvimento do ambiente de teste foi o sombreamento das malhas poligonais. Este é o processo que dá a aparência dos objetos encontrados no ambiente simulado. Inicialmente se buscou usar técnicas mais populares, como aquelas baseadas em interpolação. Estas definem a aparência das superfícies das malhas poligonais por meio de interpolações dos valores de iluminação calculados para os vértices. Tanto o valor de iluminação em cada vértice quanto o método de interpolação são definidos por meio de modelos de reflexão da luz.

A figura 15 demonstra o uso do método de sombreamento de Gouraud nas malhas poligonais de uma cadeira e das paredes do ambiente. É possível perceber na figura 15b, referente às paredes, que há descontinuidade na iluminação na aresta que liga as parendes. No caso da cadeira, a aparência obtida foi julgada insuficiente considerando o objetivo desejado. Por isso, procurou-se outros tipos de técnicas.

(a) Cadeira (b) Paredes

Figura 15 – Aplicação do método de sombreamento de Gouraud.

Em uma busca no fórum de usuários encontrou-se a implementação da técnica de sombreamento chamada *per-pixel lighting*. Como o próprio nome diz, nesta técnica o sombreamento é calculado para cada pixel. A fim de testar esta técnica, modificou-se a aplicação de teste para que esta incluísse a implementação encontrada. Na figura 16 é possível ver os resultados do teste. Observando a figura é possível perceber a continuidade da iluminação entre as paredes. Também é possível notar uma significativa melhora na qualidade na aparência da cadeira. Comparando os resultados obtidos, com aqueles demonstrados na 15, fica clara a superioridade do método *per-pixel lighting*.

Com isso, decidiu-se utilizar este método para sobreamento das malhas poligonais.

(a) Cadeira (b) Paredes

Figura 16 – Aplicação do método *per-pixel lighting*.

b) Geração de sombras

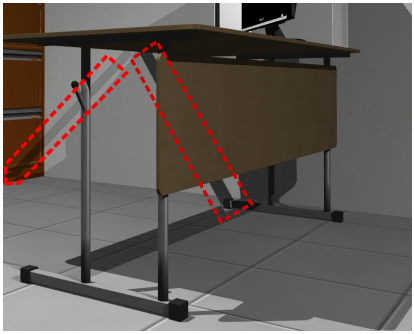
A geração de sombras foi outro aspecto da área gráfica que precisou ser solucionado. Este foi o principal problema identificado na avaliação da qualidade gráfica dos ADRs. Segundo o manual do OGRE, a criação de sombras é um dos aspectos mais desafiadores do processo de renderização 3D.

O estudo do manual revelou três meios de criar sombras no simulador. Dois deles são implementados pelo próprio renderizador, e por isso, não exigem um elevado esforço de desenvolvimento. O primeiro método é o de sombras volumétricas, também chamado de sombras *stencil* devido a tecnologia usada para implemetá-lo (*buffer stencil*). Este buffer pode ser representado como uma manto que recobre todas as superfícies da cena. Este método de sombra usa o buffer como uma máscara para fazer com que as áreas de sombra não sejam renderizadas.

Ele é chamado de sombra volumétrica pois o buffer *stencil* é preenchido por meio da extrusão da silhueta dos objetos na direção do feixe de luz incidente. Isso o torna dependente das malhas poligonais dos objetos da cena. As regiões das superfícies dos objetos que estejam em intersecção com algum destes volumes terão seu espaço no buffer marcado para não serem renderizados.

A maior vantagem deste método é que ele permite *self-shadowing*, isto é, que um objeto jogue sombras sobre si próprio. Por outro lado, ele tem a desvantagem de se tornar caro computacionamente ao aumentar o detalhamento das malhas poligonais. Outra desvantagem é o pouco realismo das sombras geradas por este método. Ao testar este método, muitos problemas foram encontrados, como pode ser observado na figura 17. Os defeitos observados na figura aparecem devido a existência de falhas nas malhas poligonais, que criam "buracos", os quais alteram a silhueta do objeto. Devido a dificuldade de corrigir todos os defeitos, o uso deste método foi descartado.

(a) Cadeira



(b) Mesa

Figura 17 – Uso de sombras volumétricas. Defeitos da imagem b marcados em pontilhado vermelho.

O segundo método de geração de sombras disponibilizado internamente pelo OGRE é o de sombras por textura. Neste método, as sombras são criadas testando se dado pixel está, do ponto de vista da fonte de luz, sob oclusão. Exitem várias maneiras de implementar esta técnica. O OGRE implementa um método bastante simples, o qual não permite *self-shadowing*, o que por si só, seria motivo para descartar este método. Mesmo assim, esta técnica foi testada, com os resultados disponibilizados na figura 18. Nos testes, ficou óbvia a impossibilidade de usar este método, devido a elevada pixelização das sombras, o que pode ser visto em ambos os exemplos.

A terceira maneira de geração de sombras, na realidade não cria sombras, na verdade ela permite que se desenvolva métodos de criação de sombras por textura mais complexos, deixando que as sombras sejam integradas ao processo de sombreamento dos objetos. Ou seja, neste método é necessário implementar conjuntamente o sombreamento e a geração de sombras. Novamente se buscou por implementações no fórum de usuários do OGRE. Felizmente foi encontrada uma implementação que junta a técnica de som-

(a) Cadeira (b) Mesa

Figura 18 – Uso de sombras por texturas.

breamento *per-pixel lighting* e o método VSM (*Variance Shadow Mapping*) de geração de sombras por textura.

(a) Cadeira (b) Mesa

Figura 19 – Uso da técnica *Variance Shadow Mapping*.

Esta técnica VSM utiliza filtragem para eliminar o *aliasing* característico de técnicas de criação de sombra baseadas em textura. Na figura 19 é possível ver o resultado do uso da implementação encontrada. É perceptível o esmaecimento das bordas das sombras, o que melhora significativamente a qualidade das mesmas. Ainda foi realizada uma modificação na implementação encontrada, para simplificar a criação de materiais a parir do material base.

Com estas considerações quanto aos aspectos visuais do simulador implementado, encerra-se este capítulo, que trata da construção do ambiente de testes. Foi apresentada uma avaliação de ferramentas para geração do ambiente, e os motivos que levaram ao criação de um *framework* para o desenvolvimento de simulações. Também foram expostos os aspectos técnicos relativos a elaboração e funcionamento deste. A seguir, será abordada a implementação de um algoritmo de visão computacional no contexto do simulador aqui descrito.

4 GERAÇÃO DE BASES DE DADOS PARA SISTEMA DE RECONHECIMENTO DE OBJETOS

Neste capítulo são abordados os aspectos teóricos e de implementação de um sistema de visão baseado no descritor de características locais SIFT (*Scale-Invariant Feature Transform*). Inicialmente é apresentado o funcionamento do SIFT, através de suas duas etapas. Em continuação, é descrito o algoritmo de reconhecimento de objetos baseado no SIFT, proposto em Lowe (2004). Na sequência, são feitas ponderações sobre este sistema e sua aplicabilidade para o problema de robótica móvel. Depois, é apresentado o algoritmo para construção de base de dados baseado em agrupamento tridimensional de poses de Lowe (2001), mostrando também as variáveis que o afetam. Por último se dá alguns detalhes sobre as implementações necessárias.

4.1 ALGORITMO SIFT

Como afirmado no capítulo 2, o SIFT é um algoritmo de descrição de características locais, o qual foi inicialmente proposto por Lowe (1999). A principal contribuição deste, é fornecer um método de descrição invariante a transformações de escala. Segundo Morel (2009), este algoritmo é baseado em um processo de normalização e simulação dos parâmetros que definem as possíveis transformações no plano. No caso, normaliza-se cada característica quanto à translação nos dois eixos, a rotação, e simula-se o parâmetro de escala. A sequência das operações do SIFT pode ser dividida em dois estágios principais:

Detecção: Responsável pela construção do espaço de escalas, localização e filtragem das características, também chamadas de pontos-chave (do inglês, *keypoint*).

Descrição: Responsável pela atribuição de orientações e construção dos descritores das características.

Como forma de compreender este algoritmo, foi elaborada sua implementação através do *Matlab*. Esta implementação foi desenvolvida com fins didáticos, uma vez que linguagens interpretadas como a do *Matlab* não são eficientes na execução de laços. Na implementação posterior, foi usada a biblioteca *openCV*, que provê resultados com maior rapidez.

a) Detecção

A fase de detecção é a responsável por determinar em que pontos da imagem existem características adequadas para descrever a imagem em questão. Estas características precisam atender o critério de estarem aparentes, independente da escala que se observe. Exemplificando com um objeto em uma imagem: se o objeto tiver seu tamanho diminuído ou aumentado na imagem (movendo o objeto na direção do eixo da câmera), e a característica aparecer independente disto, então esta característica é dita independente da escala.

Como se deseja verificar esta condição através de uma única imagem, é necessário "simular" este efeito. Esta simulação, na realidade se trata da construção da representação da imagem em um espaço de escala. Para construir o espaço de escala o autor indica uso da função gaussiana de duas dimensões:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(x^2+y^2\right)/2\sigma^2} \tag{4.1}$$

onde σ é o fator de escala. O uso desta função se deve à prova formal, desenvolvida no trabalho de (LINDEBERG, 1994 apud LOWE, 1999). Aponta-se que a função gaussiana é o único núcleo possível para análise em espaço de escalas. A construção do espaço de escala se dá através de uma série de amostragens e convoluções com a função 4.1. O resultado é uma pirâmide de imagens divida em oitavas e escalas. As oitavas se diferenciam por um fator de escala de dois, significando que a primeira escala tem o equivalente ao dobro da resolução da segunda. As escalas são as amostras intermediárias dentro de cada oitava. A figura 20 apresenta as escalas contidas na segunda oitava de uma imagem obtida no simulador.

Figura 20 – Segunda oitava obtida na contrução do espaço de escalas

Antes de buscar características, ainda é necessário uma etapa. Foi demonstrado por Mikolajczyk (apud LOWE, 2004), que pontos de máxima e mínima no laplaciano da Gaussiana normalizado pelo fator de escala ($\sigma^2 \nabla^2 G$), produzem as características mais estáveis. Lowe mostra a possibilidade de usar a diferença entre as Gaussianas (DeG) como uma aproximação do $\sigma^2 \nabla^2 G$. Desta forma, os pontos com características adequadas (pontos-chave), são um subconjunto dos máximos e mínimos da diferença entre as escalas da piramide de imagens.

Após a obtenção dos pontos de máxima e mínima, a definição dos pontos-chave é feita através de um processo de filtragem em nível de subpixel. Para isso é feita a aproximação de Taylor de segunda ordem da diferença de gaussianas, em torno do ponto de máxima ou mínima. A partir desta são calculados alguns parâmetros para verificar a robustez do ponto em questão. Primeiro, verifica-se o deslocamento do ponto de máxima da aproximação de Taylor em relação ao ponto original. Se o deslocamento em qualquer uma das direções (x,y,σ) for superior a $0,5$, então o ponto é eliminado, porque se este ponto realmente existisse, ele teria sido detectado no pixel adjacente, mais próximo da posição onde foi encontrado o pico na aproximação de Taylor.

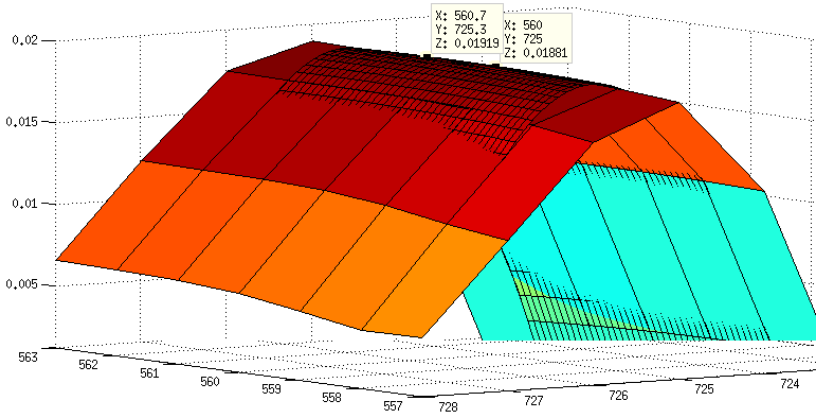


Figura 21 – Aproximação de Taylor em um ponto de máxima mal-definido

Outro parâmetro analisado é o contraste da região. Se o valor de pico da série de Taylor for inferior a $0,03$ (numa escala de 0 a 1), então há grande chance do ponto ter surgido de ruído, e por isso deve ser eliminado. Uma outra condição indesejável, são os pontos sobre bordas de objetos. As bordas aparecem fortemente na diferença de Gaussianas, e ao serem afetadas por ruídos geram pontos de máxima que são instáveis. Eles são caracterizados

por uma grande curvatura transversal e uma pequena ao longo da borda.

A razão entre as derivadas que caracterizam as curvaturas é usada para determinar se o ponto em questão está em uma borda, as quais obtidas por meio do Hessiano no ponto em questão. O ponto (x=560,y=725), avaliado na figura 21 foi eliminado devido ao deslocamento, mas também teria sido eliminado em qualquer dos testes posteriores. Um ponto de máxima ou mínima que tenha passado nestes testes é considerado um ponto-chave que deve ser descrito.

b) Descrição

A partir desta etapa, todas as operações são feitas usando a posição de cada ponto-chave como referência. Com isso garante-se a invariância do descritor quanto a deslocamentos. Dado o conjunto de pontos-chave, o estágio de descrição começa definindo a orientação de cada um deles. Isto é necessário para que se possa construir o descritor relativamente à esta orientação, tornando-o invariante a esta.

Para calcular a orientação usa-se a imagem da piramide, cuja escala é mais próxima daquela do ponto-chave. Aplica-se uma gaussiana nesta imagem, com fator 1.5 vezes a escala do ponto-chave, sobre a posição do mesmo. Isso forma uma janela, selecionando os pixels que contribuem para a aparência na região. Com os pixels desta janela, constrói-se um histograma de orientação. A orientação de cada pixel é ponderada pela magnitude deste, na janela gaussiana, a qual também faz com que os pixels mais próximos do ponto-chave tenham uma contribuição maior na definição da orientação.

O histograma de orientações é formado por 36 células, onde deve ser encontrada aquela com maior valor. Se existirem outras células com mais de 80% do valor do pico, elas são marcadas como orientações adicionais. Até quatro orientações podem ser armazenadas por ponto-chave. Efetivamente esta condição permite a criação de até quatro descritores para o mesmo ponto-chave. Para finalizar a definição da orientação, é feito o ajuste de uma parábola sobre as duas células adjacentes a cada orientação encontrada. Isso permite a interpolação da região do histograma, fornecendo um valor de orientação mais preciso.

A última etapa é a construção do descritor propriamente dito. O descritor SIFT é basicamente um histograma tridimensional que acumula os gradientes dos pixels da região em torno ao ponto-chave. Estes funcionam como as amostras, as quais tem sua magnitude acumuladas, indexando a localização espacial (x,y) do pixel e a orientação do gradiente. Para compensar a orientação do ponto-chave são rotacionados os índices espaciais e a orientação do gradiente.

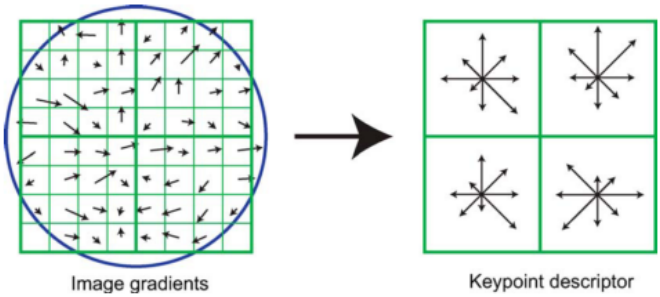


Figura 22 – Construção do descritor SIFT. O círculo representa a abrangência da janela gaussiana. As flechas na parte da esquerda representam os gradientes. Descritor na direita com 2x2x8 células. Fonte:Lowe (2004)

A magnitude do gradiente é ponderada por uma janela gaussiana, análoga a usada para o cálculo da orientação. Porém neste caso o σ é metade da largura total da região amostrada pelo descritor. Cada célula espacial do histograma têm um tamanho de 4x4 pixels, o que permite que uma amostra de gradiente mude até 4 posições sem que se altere o descritor. O descritor SIFT padrão é obtido a partir de uma região de 4x4 células espaciais, as quais possuem 8 células de orientação cada, formando um descritor de dimensão 128.

Na figura 22 é possível ver a representação do processo de construção do descritor usando uma região de 2x2 células espaciais. Para limitar os efeitos de borda é usada uma interpolação trilinear para "espalhar" a contribuição de uma amostra para as células adjacentes. Para mitigar os efeitos de mudanças na iluminação, o descritor é normalizado. A aplicação do algoritmo completo em uma figura gerada no simulador é apresentada na figura 23, onde o número de pontos-chave foi limitado a 50 para melhor visualização.

4.2 SISTEMA DE RECONHECIMENTO DE OBJETOS BASEADO NO SIFT

Feita a descrição do SIFT, pode-se apresentar o sistema de reconhecimento de objetos baseado neste, apresentado em sua versão final em Lowe (2004). Este sistema, fundamenta-se em encontrar uma transformação afim que descreva o mapeamento dos parâmetros de um subconjunto de pontos-chave da imagem de amostra, e um subconjunto contido em um modelo da base de dados. Perceba que neste sistema os modelos da base de dados são

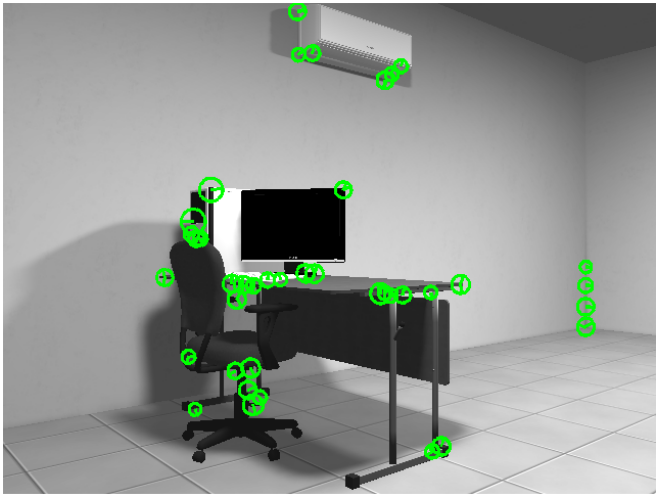


Figura 23 – Uso do SIFT sobre uma imagem do simulador.

formados apenas por descritores SIFT e os parâmetros de localização, orientação e escala respectivos. Para encontrar a transformação, o sistema usa uma estratégia baseada em votação. Ele é formado pelas seguinte etapas: **Correspondência** entre pontos-chave, **Agrupamento** dos mesmos, **Definição da hipótese** por solução de parâmetros afins e aplicação de modelo probabilístico para aceitação da hipótese.

a) Correspondência entre amostra e base de dados

Primeiramente, são levantadas as correspondências entre a imagem de amostra e cada modelo na base de dados. As correspondências são encontradas através do cálculo da distância euclidiana entre as características da imagem de amostra e o modelo. Porém é necessário tratar a possibilidade de um ponto-chave da amostra ser espúrio. Estes podem ter origem no ruído de fundo das imagens, ou em descritores não existentes na base de dados. Para isto é utilizado um limiar baseado na razão entre a menor e a segunda menor distâncias obtidas.

O motivo para usar esta abordagem em detrimento ao uso de um limiar global, está no fato dos pontos-chave possuírem diferentes graus discriminatórios. Pontos-chaves espúrios tendem a gerar distâncias muito próximas quando comparados com a base de dados, enquanto em pontos-chaves reais, a comparação de suas informações tendem a encontrar apenas uma característica com propriedades próximas. Isso faz com que a distância obtida para

essa única característica seja bastante inferior as distâncias obtidas em relação a outras. Para base de dados grandes, o autor sugere o uso de uma abordagem baseada em uma técnica de vizinho-mais-próximo. No contexto deste trabalho não foi analisada esta questão, porque inicialmente o desempenho temporal não é a maior preocupação.

b) Agrupamento de características

Com o conjunto de pares de características SIFT correspondentes definido, avança-se para criação das hipóteses de transformação. Devido a possibilidade de ocorrer oclusão, ou do objeto ser pequeno, é necessário que o sistema de reconhecimento seja capaz de funcionar com poucas características disponíveis. Outro complicador nesta etapa, é a existência de falsas correspondências no conjunto obtido na etapa anterior.

O limiar usado para descartar correspondências de ruído, não elimina completamente falsas correspondências. Assim, torna-se necessário identificar o subconjunto de correspondências corretas dentro do conjunto total. Isso pode vir a exigir a eliminação de 99% das correspondências encontradas. Esta condição impede o uso de métodos de ajuste como o RANSAC, uma vez que estes não têm um bom comportamento quando o número de *outliers* é maior que 50%.

A solução encontrada foi utilizar a transformada generalizada de Hough (BALLARD, 1981). Este algoritmo identifica os grupos de correspondências que têm uma interpretação semelhante quanto ao mapeamento entre modelo e imagem de amostra. O algoritmo fundamenta-se na discretização do espaço de parâmetros de transformação, através da construção de células para cada região do espaço. Este novo espaço é chamado de espaço de Hough. Sobre este, cada correspondência "vota" na célula que representa os parâmetros obtidos do mapeamento de seu par de características. No caso do SIFT, o espaço de parâmetros é de quatro dimensões (escala,translações e rotação no plano).

A estratégia usada pela transformada de Hough funciona porque apenas as correspondências corretas tendem a concordar quanto uma possível transformação. Como as falsas correspondências não derivam do casamento correto com um modelo, estas tendem a espalhar seus votos pelo espaço de Hough, devido a sua natureza randômica. Uma vez computados os votos, basta encontrar a células que possuem votos em maior quantidade. O autor do SIFT afirma que é possível obter uma hipótese válida verificável com pelo menos três correspondências concordantes. Assim, qualquer célula com três ou mais votos, representa uma hipótese de mapeamento.

c) Definição do mapeamento e decisão

Os resultados obtidos a partir da transformada de Hough apenas limitam as regiões do espaço de parâmetros onde pode-se encontrar o mapeamento entre os pontos-chave da imagem de amostra e do modelo. Para se determinar este com precisão, é necessário submeter as hipóteses de Hough a uma verificação geométrica. Faz-se isto definindo os parâmetros da transformação afim que faz o mapeamento entre os o pares de pontos formadores das correspondências geradoras da hipótese. A transformação afim de um ponto do modelo $[x\ y]^T$ em um ponto da imagem $[u\ v]^T$ é denotada por:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{4.2}$$

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ x & y & 0 & 0 & 1 & 0 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix} \Leftrightarrow \mathbf{Ax} = \mathbf{b} \tag{4.3}$$

onde $[t_x\ t_y]^T$ representa a translação do modelo, e m_i representam os parâmetros de escala, rotação e cisalhamento. Isolando os termos desconhecidos de 4.2, obtêm-se o sistema 4.3, no qual só um ponto chave está representado

Resolve-se este sistema por meio de mínimos quadrados, o que minimiza a soma dos quadrados das distâncias entre a projeção dos pontos do modelo e suas correspondências na imagem. Verifica-se a concordância dos pontos com a solução encontrada de maneira iterativa. São descartados os pontos para os quais o mapeamento gere um erro maior que a metade das dimensões das células de Hough. Depois da exclusão dos pontos discordantes os parâmetros da transformação são recalculados. O processo é encerrado caso restem menos que três pontos ou que todos concordem com o mapeamento. No primeiro caso, a hipótese é descartada. Caso restem hipóteses, estas são avaliadas em um modelo probabilístico baseado em inferência bayesiana.

4.2.1 Análise do algoritmo no contexto da robótica móvel

Considerando os conceitos apresentados na seção 2.1 e aspectos relacionados à visão, foi feita uma análise do algoritmo proposto por Lowe (2004). Do ponto de vista de execução de tarefas, a principal propriedade do sistema apresentado é a capacidade de, uma vez reconhecido o objeto, conse-

guir informar os parâmetros que permitem localizar o objeto no ambiente.

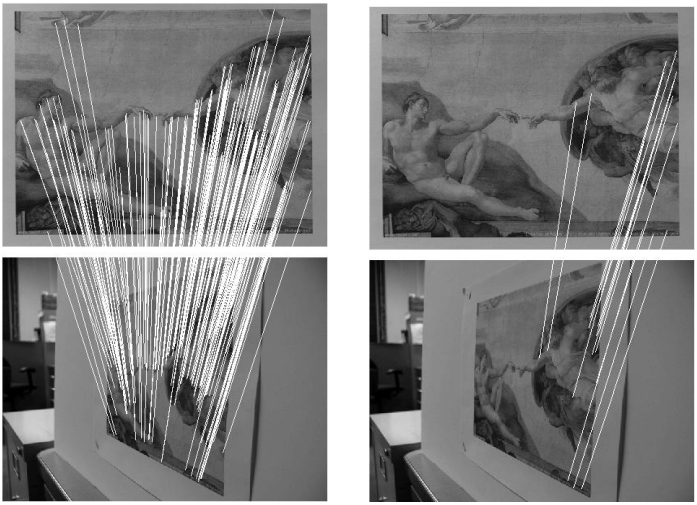
Tomando isto como premissa, fica clara a utilidade deste método na execução das tarefas ligadas à movimentação, principalmente em tarefas de localização. Se o objeto localizado for estático no ambiente, pode-se deduzir a posição e orientação atual do robô. No caso de tarefas de mapeamento este sistema pode ser aplicado na construção de mapas topológicos que indexam objetos. Ele ainda poderia substituir o sistema de reconhecimento presente no trabalho de Kim e Nevatia (1999), tornando-o útil em tarefas de navegação.

Com relação aos conceitos ligados a algoritmos de movimentação, é preciso considerar que nem sempre o sistema conseguirá reconhecer os objetos do ambiente, e que é possível ocorrerem falsos positivos. Por isso, sistemas de movimentação que exijam níveis elevados de completude e otimalidade precisariam de alternativas para cobrir eventuais falhas do sistema de reconhecimento de objetos. Levando em conta que o tempo de computação do algoritmo é da ordem de um segundo, pode-se afirmar que ele é aplicável a métodos de movimentação *online*. Quanto a complexidade, esta é limitada ao número de pontos-chave na base de dados e a quantidade dos mesmos nas imagens de amostra.

Já no contexto de visão computacional percebe-se uma limitação do algoritmo a qual deve ser observada. Baseando-se no SIFT, ele têm acesso a características invariantes a mudanças de escala, rotação e translação. Devido ao método de construção do descritor adquire-se alguma robustez a mudanças na pose do objeto. É justamente neste ponto que o algoritmo pode não se comportar adequadamente. Morel (2009) demonstrou as limitações do SIFT ao compará-lo com sua própria proposta de descritor (ASIFT *Affine-SIFT*). A figura 24 demonstra a comparação entre as duas técnicas, em uma imagem obtida em uma pose com ângulo frontal de 75°. Através do SIFT foi possível obter apenas 15 pontos-chave, enquanto o ASIFT forneceu 202.

Nesta situação, o sistema de reconhecimento de objetos usando o SIFT, teria grande dificuldade de fornecer informações relevantes. Devido a sua estratégia baseada em votação, a existência de poucos pontos pode fazer que não emerjam hipóteses para identificação do mapeamento, o que impediria o reconhecimento. A escolha pelo ASIFT está atualmente limitada as implementações existentes. Pinage et al. (2012) demonstra em sua pesquisa sobre detecção de marcações naturais para navegação de um VANT(veículo aéreo não tripulado) que o ASIFT é cerca de dez vezes mais lento que o SIFT, precisando, em média, cerca de 10 segundos para processar uma imagem.

A maneira de contornar a situação com relação ao SIFT é usar uma base de dados com múltiplas poses de cada objeto. Assim, espera-se que ao encarar que uma situação como a da figura 24b, encontre-se uma pose mais próxima que permita o reconhecimento. Porém, esta abordagem também tem



(a) ASIFT (b) SIFT

Figura 24 – Aplicação de descritores de características a uma imagem obtida em um ângulo de 75°. Fonte:Morel (2009).

um ponto-fraco. Ao se comparar uma imagem de amostra com uma base de dados com múltiplas poses deste, ocorre uma dispersão das correspondências entre as imagens. Esta condição causa o mesmo efeito que se desejava inicialmente eliminar. Para sobrepujar este problema, Ramisa et al. (2009) indicou o uso da proposta feita por Lowe (2001).

4.3 ALGORITMO PARA GERAÇÃO DE BASE DE DADOS

No algoritmo de reconhecimento de objetos original, cada imagem representa um modelo. Nesta proposta de Lowe (2001), agrupa-se imagens de treinamento obtidas de poses semelhantes em um modelo de pose. Desta forma, o modelo de um objeto consiste de um conjunto de modelos de pose que representam a aparência do mesmo a partir de poses significativamente diferentes.

O processo de construção da base de dados inicia-se tomando a primeira imagem de treinamento como modelo inicial. Todos os descritores de pontos-chave são adicionados a base de dados, armazenando também a localização, rotação e escala destes. A partir daí, usa-se uma abordagem parecida com a usada para o reconhecimento: para cada nova imagem de treinamento,

usa-se a transformada de Hough sobre o conjunto de correspondências encontradas, e uma subsequente análise geométrica para determinar se há algum modelo de pose equivalente à imagem de treinamento corrente.

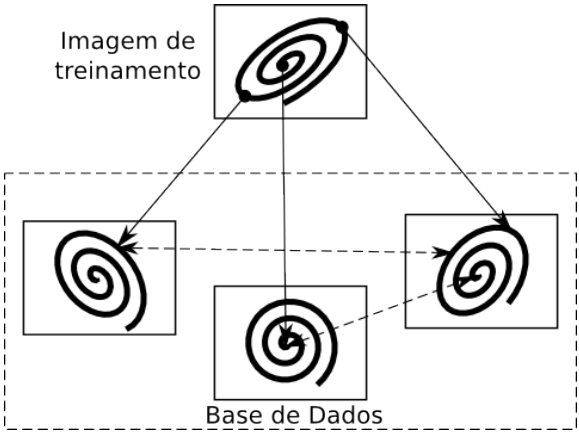


Figura 25 – Ligação entre poses diferentes do mesmo objeto

Com o agrupamento, características em cada modelo de pose são conectadas às outras semelhantes por meio de ligações (*links* em inglês). Ao se testar uma nova imagem, seus pontos-chave irão corresponder com características de vários modelos de pose (figura 25). Seguindo as ligações destas correspondências, pode-se acumular as correspondências em outros modelos de pose, evitando que ocorra o problema de dispersão comentado anteriormente.

A maior diferença em relação ao algoritmo de reconhecimento de objetos está no uso da transformação de similaridade para criação das ligações, ao invés da transformação afim. Assim, reescreve-se o sistema 4.2 da seguinte forma:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m & -n \\ n & m \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \cdots \Leftrightarrow \mathbf{Ax} = \mathbf{b} \tag{4.4}$$

onde o sistema 4.4 é resolvido por mínimos-quadrados, da mesma forma que 4.3.

Com a solução **x**, obtida a partir das correspondências que concordaram com uma mesma hipótese no espaço de Hough, estima-se o erro **e** da projeção de todas as correspondências obtidas entre o modelo e a nova ima-

gem:

$$\mathbf{e} = \sqrt{\frac{2\left\|\mathbf{Ax}-\mathbf{b}\right\|^2}{r-4}}$$

onde r é o número de linhas da matriz $\mathbf{A}$, do qual se subtrai os 4 graus de liberdade da transformação de similaridade. O fator 2 refere-se ao fato de cada correspondências ocupar duas linhas na matriz $\mathbf{A}$, e portanto seus erros devem ser somados para se mediar a distância quadrática entre a imagem e o modelo.

Sobre o erro $\mathbf{e}$ toma-se a decisão final sobre o agrupamento, ou não, da nova imagem de treinamento. Aplica-se um limiar T, igual a $0,05$ vezes a dimensão (x,y) máxima da imagem testada. O autor afirma que esse limiar é suficiente para agrupar imagens com diferenças de até 20 graus em rotações de profundidade. Considerando isto, há três possibilidades quanto ao agrupamento de uma nova imagem de treinamento:

1. A imagem de treinamento não possui correspondências na base de dados. Portanto deve iniciar um novo modelo de objeto

2. A imagem de treinamento possui correspondências com um erro de projeção $e > T$. Esta deve ser adicionada como um novo modelo de pose do modelo de objeto correspondente. Este modelo é criado de maneira similar a um novo modelo de objeto, porém, são adicionados a este as ligações aos modelos de pose mais próximos.

3. A imagem de treinamento possui correspondências, com um erro de projeção $e \leq T$. Neste caso, esta imagem é combinada ao modelo de pose mais próximo, projetando todas as sua características para o modelo existente usando a transformação de similaridade obtida. Além disso, armazena-se as ligações destas novas características com até três outros modelos de pose mais próximos. Elimina-se as características que forem muito próximas de outras já existentes.

4.3.1 Variáveis da geração de bases de dados

A partir do algoritmo de agrupamento de poses apresentado, deseja-se criar bases de dados úteis em aplicações de reconhecimento de objeto em robótica móvel, e para tanto, devem ser avaliados alguns aspectos do processo de construção destas. Para que seja possível fazer o reconhecimento com um sistema como o da seção 4.2, é necessário ter um conhecimento anterior

sobre objetos, o que induz a criação *offline* das bases de dados. Para tanto, duas variáveis são notáveis: a maneria de **amostrar** o objeto e **construir as ligações**.

a) Amostragem

Em um processo de geração offline das bases de dados, pode-se escolher a amostragem do objeto de acordo o desejado. Desta forma é ela que define o número de imagens obtidas por objeto, o que impacta diretamente na quantidade de informação armazenada na base de dados. O modo mais simples de amostrar a aparência de um objeto é capturar imagens em intervalos constantes de deslocamento em torno do objeto. No caso de um objeto tridimensional, isto significaria percorrer o "equador" de uma esfera que o circunscreve, e capturar as imagens.

Outra forma simples seria utilizar uma proposta da área de reconhecimento de formas (CHEN et al., 2003), que circunscreve o objeto com polígonos regulares, como dodecaedros, e obter imagens a partir do centro de suas faces (figura 26). Esta abordagem tem o diferencial de tomar imagens em duas alturas opostas, uma negativa em relação a linha de centro do objeto, e outra positiva.

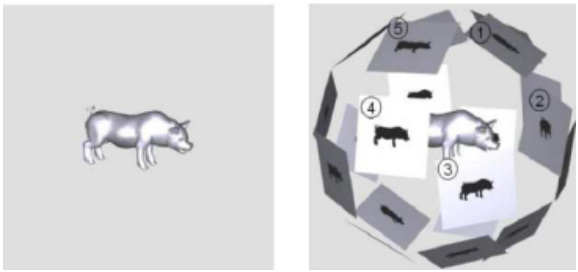


Figura 26 – Imagens capturadas das faces de um dodecaedro. Na esquerda o objeto, na direita a captura das imagens. Fonte: Chen et al. (2003)

Uma abordagem mais complexa seria fazer uma amostragem não uniforme, para privilegiar regiões do objeto com mais informação. Porém, seria necessário fazer primeiro a amostragem uniforme, para posteriormente usar algum método para mensurar a contribuição de cada amostra e inferir a região de maior interesse. Como seria necessário fazer a amostragem uniforme antes, foi decidido não avaliar esta hipótese neste trabalho.

b) Construção das ligações

Lowe (2001) deixa implícito na descrição do algoritmo o fato das ligações serem bi-direcionais. Isto torna necessário que as ligações geradas por novos modelos devam ser copiadas para os modelos ligados, com os indexadores invertidos. Como o processamento é feito sequencialmente, esta é a única maneira de um modelo já existente receber novas ligações. Tal característica pode causar algumas situações anormais, considerando que, ao criar um novo modelo de pose, ele só poderá ser ligado a até três outros modelos de pose existentes.

Do ponto de vista de um modelo anterior, pode acontecer que o novo modelo de pose seja mais semelhante que os outros disponíveis até o momento. Porém existem outros três modelos semelhantes entre si, que ocupam as três ligações do modelo mais recente. Desta forma o referido modelo antigo não estaria ligado ao seu modelo de pose mais próximo. Outro ponto importante, é que esta abordagem não limita que múltiplos modelos de pose se liguem a um mesmo modelo. No limite, um modelo de pose pode monopolizar as ligações, estando ligado à todos os outros modelos de pose do objeto.

Se o modo de construção das bases fosse *online*, seria mandatório usar esta abordagem para criação de ligações, pois seria necessário inverter as ligações assim que estas fossem criadas. Como no caso a abordagem é *offline*, a criação das ligações reversas pode ser feita no final do processo. Além disso, o uso desta abordagem permite um o uso de uma nova estratégia para criação dos ligações.

Como já se tem todas as imagens usadas na sequência em que foram capturadas, propõe-se a abordagem de re-executar o algoritmo, porém com a sequência de imagens invertida. Desta maneira, que cada modelo sempre se ligará ao modelo mais próximo possível, e também limitará um modelo de pose a ter no máximo seis ligações.

Porém a maior diferença está no fato que esta abordagem não garante que as ligações sejam bidirecionais. Elas podem eventualmente ser bidirecionais, caso os modelos de pose ligados sejam mutuamente os mais próximos. Entende-se que a bidirecionalidade não é um pré-requisito para o funcionamento do agrupamento, uma vez que o objetivo é apenas ligar os modelos mais próximos entre si. Desta forma acredita-se que seja factível a utilização do recálculo das ligações

4.4 IMPLEMENTAÇÕES DESENVOLVIDAS

Para a avaliar as variáveis que poderiam afetar a geração de base de dados, foi feita a implementação do algoritmo de agrupamento de poses, apre-

sentado na seção 4.3, usando o *framework* apresentado no capítulo 3. A única biblioteca utilizada foi a *openCV*, que forneceu os métodos matemáticos, estruturas de dados e a implementação do SIFT. Elaborou-se a estrutura do código de forma a permitir a execução automática de testes, onde os parâmetros são informados por meio de um arquivo *xml* de configuração, e os resultados através de arquivos *xml* e *yml*.

Levando em consideração a estrutura do *framework*, foi criada a classe `DataBaseGen` para representar uma entidade do simulador, para ser responsável pela movimentação da câmera e pelo posicionamento e controle dos objetos a serem adicionados à base de dados. Para executar a captura da imagem, foi utilizada uma abordagem onde o objeto fica estacionário e centrado na origem do ambiente, com a câmera percorrendo uma lista hierárquica de poses, permanecendo sempre apontada para o centro do objeto.

Estas poses são definidas por meio de dois ângulos: altitude e azimute. Executa-se todos os azimutes listados sob uma determinada altitude, podendo estes serem arbitrários tanto em quantidade quanto valor. Existem listas pré-definidas de poses para testes mais comuns. Para outros valores usa-se o arquivo *xml* definindo a lista de altitudes e a lista de azimutes subordinada a cada altitude desejada. A distância da câmera ao objeto é definida automaticamente, de forma que o volume delimitante seja completamente visível de qualquer posição de uma esfera que circunscreva o objeto sendo testado.

Os objetos a serem reconhecidos são carregados na cena a partir dos recursos disponíveis no renderizador. Todos os objetos cujas malhas poligonais possuem o prefixo "`o_`" no nome, são usados na construção do ambiente e marcados para fazer parte da base de dados. Toda a vez que a lista de poses é percorrida completamente, a entidade da classe `DataBaseGen` remove o objeto presente e adiciona o próximo.

Além destas atribuições, esta entidade ainda é responsável por instanciar a tarefa `DataBaseGenTask`, que efetivamente constrói a base de dados. Nesta classe é implementado o algoritmo de agrupamento tridimensional de poses da seção 4.3, que recebe como entrada, os parâmetros do descritor SIFT. Para garantir que apenas os pontos-chave pertencentes ao objeto sejam adicionados à base de dados, usou-se um recurso do simulador para traçar raios a partir da câmera, no sentido dos pontos chaves detectados. Se o raio intersecta o objeto, então este é adicionado a base de dados. Caso contrário é descartado.

A saída é fornecida em cinco arquivos *yml*, que permitem uso facilitado para utilização posterior, por métodos do *openCV*. Para cada modelo de pose da base de dados, os arquivos contém o conjuntos de pontos-chave, descritores SIFT, ligações para outros modelos e contadores referentes a estes. Os arquivos se diferenciam na origem da informação. Há um arquivo

com o resultado apenas da operação sequencial normal, e outros dois arquivos armazenam exclusivamente os resultados da inversão das ligações e do reprocessamento em ordem inversa. Os dois últimos fornecem a união dos resultados da operação normal, tanto com a inversão das ligações, quanto com o reprocessamento em ordem inversa.

Como recurso necessário para a elaborar o sistema de geração de base de dados, foi preciso desenvolver uma implementação da Transformada Generalizada de Hough para modelos do SIFT, pois esta não está disponível. Ela foi elaborada com base na proposta original de Ballard (1981) e em Nixon e Aguado (2008). O método `houghTransform` cria o espaço de Hough com as quatro dimensões $[x\, y\, \theta\, \sigma]$ que definem os modelos de objeto. Este espaço é feito através de matrizes esparsas, que armazenam instâncias da estrutura `HoughSpaceBin`. Esta armazena as correspondências que votaram para determinada célula, e os parâmetros dos espaço de Hough que ela representa.

Com esta descrição das implementações necessárias, finaliza-se este capítulo, onde foram apresentados os algoritmos relacionados à geração de bases de dados para um sistema de reconhecimento de objetos baseado no SIFT. Foram feitas considerações sobre a aplicabilidade do sistema de reconhecimento para problemas de robótica móvel e as variáveis do algoritmo de agrupamento de poses que afetam a geração da bases de dados. Na sequência, serão apresentados os resultado obtidos nesta pesquisa de mestrado.

5 RESULTADOS

Finalizada a exposição das etapas de desenvolvimento deste trabalho, passa-se para a apresentação dos resultados alcançados. A descrição dos resultados fará, sempre que possível, referência aos objetivos listados no capítulo 1, ou as seções dos capítulos 3 e 4. Os objetivos específicos serão apresentados inicialmente, e posteriormente será feito o juízo quanto aos objetivos gerais. Para facilitar a compreensão, a tabela 6 reapresenta os objetivos, com um resumo dos resultados associados.

Tabela 6 – Relacionamento entre objetivos e resultados

Objetivos		Resultados Associados
1	**Simular robô**	Simulação comprovada através dos testes realizados.
1.1	Análisar ADRs existentes	Qualidade gráfica padrão não suficiente em ambos. Implementação da solução seria mais complicada.
1.2	Implementar o ambiente	Criado um escritório com 13 objetos, conforme demonstrado na figura 27.
		Framework criado pode ser usado em outros trabalhos.
1.3	Teste de simulação	Calibração demonstrou a exatidão dos parâmetros estéreo obtidos do simulador.
2	**Gerar bases de dados**	Gerou-se bases de dados com múltiplas configurações no simulador implementado.
2.1	Implementar o sistema	Implementação apresentada na seção 4.4.
		Transformada de Hough do SIFT utilizada em outro trabalho.
2.2	Identificação das variáveis	Seção 4.3.1 apresentou as duas variáveis: a) Amostragem; b) Criação de ligações.
2.3	Avaliação	a) Ângulos acima de 30° geram dificuldade na geração de ligações. b) Recálculo distribui melhor as ligações.

Os objetivos específicos ligados ao desenvolvimento do ambiente de testes foram os primeiros a serem alcançados. No objetivo 1.1, que buscou promover a análise de ambientes de desenvolvimentos de robôs (ADRs) existentes, fez-se a avaliação dos ADRs Gazebo e MORSE através da modificação da metodologia proposta por Kramer e Scheutz (2007). Verificou-se que o maior problema com estas ferramentas foi quanto a qualidade gráfica disponibilizada pelos recursos oferecidos por padrão. Nota-se que seria possível implementar os algoritmos de computação gráfica da solução alcançada em ambos os ADRs. Porém, isso demandaria um esforço maior, pois seria necessário compreender com profundidade o código-fonte dos ADRs, para descobrir como integrar as soluções encontradas, e que já estavam implementadas de forma útil.

O segundo resultado alcançado neste trabalho está diretamente relacionado com o objetivo 1.2, de construir um ambiente para testes. Conforme demonstrado na seção 3.2 , obteve-se uma maneira de simular o ambiente de um escritório, o qual contém treze objetos fixos, de dez tipos diferentes. Na figura 27 é possível observar o resultado final obtido.

Durante o desenvolvimento necessário para alcançar este objetivo-específico, se alcançou um terceiro resultado, que foi o desenvolvimento de um *framework* para desenvolvimento de simulações tridimensionais, mostrado na seção 3.2.1. Além deste ser a pedra fundamental na implementação do simulador usado neste trabalho, este *framework* pode ser útil para testar outros algoritmos de visão computacional. Com ele é possível fazer testes em ambientes de alta-fidelidade, e em condições controladas e com alta repetitividade. Isso facilita a comparação de diferentes técnicas e a observação da variação de resultados ao se variar os parâmetros de teste. Outra vantagem é a possibilidade de realizar testes livres de ruídos comuns em testes com equipamentos reais, o que permite que este problema possa ser atacado separadamente.

Uma deficiência da implementação atual do *framework*, é não poder garantir a reprodutibilidade de testes. Como os simuladores desenvolvidos através deste *framework* fazem uso de trechos de código executados em processadores gráficos, não se pode garantir a mesma aparência em todos os computadores. Isso é derivado da maneira como são calculados o sombreamento e a geração de sombras, que são altamente dependentes das capacidades disponibilizadas pelo hardware. A figura 28 apresenta a execução do simulador com um processador gráfico de um fabricante diferente daquele usado na figura 27 (*AMD* e *nvidia* respectivamente). Note as diferenças nas nuances das sombras dos objetos e do sombreamento do teto.

Para se verificar a capacidade de simulação oferecida pelo simulador desenvolvido, como consta do objetivo 1.3, foi implementado um robô com

Figura 27 – Vista frontal e de fundo do ambiente de escritório desenvolvido através do simulador

sistema de câmeras estéreo. Cada câmera foi configurada para gerar imagens com resolução VGA (640x480), o que consequentemente definiu a razão de aspecto. Como não se tem uma lente propriamente dita, não é apropriado usar a informação de distância focal para configurar as câmeras. No lugar de usar a distância focal, foi utilizada as informações de campo de visão.

Figura 28 – Ambiente de escritório simulado em outro computador.

Considerando que a razão de aspecto já estava decidida, foi necessário definir apenas um dos ângulos. No caso, foi usado um ângulo vertical de 45° para ambas as câmeras, como o usado pelo sistema Navcam[1]. Como não se encontrou a informação de linha de base deste sistema, utilizou-se o valor de 6 (seis) centímetros, como o sistema nDepth Vision[2].

Com o intuito de executar o teste de calibração, implementou-se a tarefa `CameraCalib`. O objetivo principal desta tarefa foi informar os parâmetros intrínsecos e extrínsecos das câmeras. Para isto, capturou-se pares de imagens estéreo as quais continham um plano com um padrão quadriculado. A elaboração do código através de herança da classe `Task`, permitiu que a seleção dos pares estéreo fosse feita online. A escolha se deu por inspeção visual através de telas implementadas pela classe `Miniscreen`. Uma vez adquiridos os pares, foram usadas as funções de calibração do *openCV* para obter as matrizes de parâmetros intrínsecos e extrínsecos. Os resultados mais importantes, obtidos por meio de 18 pares de imagens estão na tabela 7.

A informação do erro RMS foi obtida diretamente da função de retificação utilizada. O erro médio de reprojeção foi calculado através do uso da restrição epipolar $x'^T F x = 0$, onde F é a matriz fundamental, para mensurar o erro médio ao projetar os pontos usados da calibração. Os resultados de-

[1] `http://marsrover.nasa.gov/mission/spacecraft_rover_eyes.html`
[2] `http://www.focusrobotics.com/products/systems.html`

Tabela 7 – Resultados da calibração

Parâmetro	**Valor nominal**	**Valor obtido**
x	-0.06 m	-0.060018 m
Translação y	0.00 m	2.4×10^{-5} m
z	0.00 m	3.9×10^{-5} m
Ângulo de visual	45°	45.0112°
	erro RMS	0.1099 px
	erro médio de reprojeção	0.0465 px

monstram, como esperado, que a definição de parâmetros de câmera dentro do simulador podem ser feitas com uma elevada exatidão. Além disto, este resultado também é importante para confirmar a possibilidade de testar um robô com câmeras no ambiente simulado, o que indica o cumprimento do objetivo 1.

Avançando para os objetivos relacionados à geração de base de dados, realizou-se inicialmente a implementação do sistema de captura de imagens e criação de bases de dados, ligada ao objetivo 2.1, o que já foi apresentado na seção 4.4. Destaca-se a implementação da transformada generalizada de Hough para o SIFT, pois esta não tem implementação disponível publicamente, além do fato de já estar sendo usada em outra pesquisa de mestrado.

O objetivo 2.2 que trata da identificação das variáveis do algoritmo de agrupamento que poderiam afetar a construção das bases de dados foi tratado na seção 4.3.1. Nesta, verificou-se a existência de duas variáveis: **amostragem** do objeto e método de **criação das ligações**. Com a definição destas, avançou-se para o objetivo específico 2.3, que objetiva a avaliação dos efeitos que a alteração destas variáveis causam nas bases de dados. Estes testes foram feitos de maneira sintética, pois ainda não se tem a implementação do sistema de reconhecimento de objetos necessário para que se possa testar estas de maneira prática.

Para definição dos parâmetros do SIFT a serem usados nos testes, foram feitos testes preliminares de geração da base de dados, amostrando-se em torno dos valores indicados por Lowe (2004). Nestes testes, o objetivo era maximizar o número de pontos-chave e minimizar do número de modelos de pose. Estas condições indicam a quantidade de informação contida na base de dados, qualificando-a de acordo com sua coesão no menor número possível de modelos. Por meio destes testes preliminares, verificou-se que os parâmetros usados por Lowe (2004) apresentavam o melhor compromisso entre as duas métricas. Desta forma, todos os testes realizados neste capítulo

foram desenvolvidos usando estes valores.

Para que as bases de dados criadas fossem úteis para um futuro uso no robô simulado no objetivo 1, foram utilizados apenas os objetos contidos no ambiente de simulação do escritório. Foram usados os dez tipos de objetos usados no ambiente mais a porta para criação da base de dados, totalizando onze objetos.

A primeira variável da criação de bases de dados testada foi a amostragem dos objetos. Este teste foi feito usando apenas a criação inversa de ligações, pois gostaria-se de observar os resultados com o método padrão. Foram testados seis ângulos de amostragem: 10°, 15°, 20°, 30°, 40°, 60°. Não foram testados ângulos maiores que 60°, devido aos resultados de Morel (2009) mostrados na seção 4.2.1, que indicavam a dificuldade do SIFT em ângulos de 75° .

Também testou-se a abordagem por polígonos de Chen et al. (2003), usando o dodecaedro. Foram amostradas somente 10 faces do dodecaedro, uma vez que os robôs não conseguiriam ver o topo ou fundo dos objetos. Para cobrir qualquer aleatoriedade, foram realizados oito experimentos para cada ângulo de amostragem usado. Os resultados do teste estão na tabela 8, indicando sempre que necessário o intervalo nos quais estão valores médios, considerando uma confiança de 95%.

Tabela 8 – Resultados do teste de Amostragem

	Ligações	**Pontos-chave**	**Ligações/Pontos**
D10[a]	1399.5 ±4	3552.0 ±2	0.39
60°	2096.2 ±2	2131.4 ±1	0.98
40°	3563.8 ±13	3102.4 ±3	1.15
30°	6895.2 ±9	4075.2 ±2	1.69
20°	12450.0 ±31	5969.0 ±3	2.09
15°	19910.0 ±38	7995.0 ±4	2.49
10°	34275.0 ±137	11818.6 ±54	2.90

	Imagens-I	**Modelos-M**	**Pontos/modelos**	**M-I(%)**
D10[a]	110	107.0 ±0	33.2	2.7%
60°	66	62.0 ±0	34.4	6.1%
40°	99	96.0 ±0	32.3	3.0%
30°	132	119.0 ±0	34.2	9.8%
20°	198	182.0 ±0	32.8	8.1%
15°	264	239.0 ±0	33.5	9.5%
10°	396	355.0 ±1	33.3	10.4%

[a]D10 : Dodecaedro - 10 imagens

M-I(%): Redução percentual de modelos (M) com relação ao total de imagens (I) capturadas

Inicialmente percebe-se que a aleatoriedade na geração das bases de dados é baixa. O pior caso ocorreu com o ângulo de 10°, onde o intervalo de confiança no número de pontos-chave representa menos de 1% da quantidade total. Também percebe-se que a estratégia com dodecaedros é claramente inferior às outras, sendo a única em que o número de ligações foi significativamente inferior ao número de pontos-chave. Acredita-se que a razão para isto está na diferença em graus de uma imagem para outra, que é de 72° para imagens na mesma altitude. A criação de ligações com os ângulos de 40° e 60° se mostrou mais difícil, conforme indicado na razão entre o número de ligações e o número de pontos. Apenas os ângulos de amostragem de 20° e inferiores conseguiram gerar em média mais de duas ligações por ponto-chave.

O critério mais importante nesta avaliação é o de redução do número de modelos. Ele indica a diminuição ocorrida ao agrupar o conjunto de imagens obtidas com determinado ângulo de amostragem. No caso, do ângulo de 10° houve uma redução da ordem de 10%, uma vez que foram construídos 355 ± 1 modelos, a partir de 396 (11 objetos vezes 36 imagens por objeto) imagens capturadas para construção desta base de dados. De modo geral esperava-se que o agrupamento de poses permitisse uma diminuição maior no número de modelos de pose.

Observando os resultados verifica-se que o motivo para isso está na baixa média de características por modelo de pose. Como há poucos pontos-chave, torna-se mais difícil que vários votos no espaço de Hough se agrupem, dificultando a geração de hipóteses. Do ponto de vista prático, a razão para isto acontecer está no fato das texturas usadas na construção do objetos serem simples e suaves. Por isso, pode-se argumentar que o ambiente simulado é a origem do problema. Porém, espera-se que um sistema de reconhecimento de objetos para robótica possa reconhecer objetos com aparência semelhante daqueles usados na simulação. De qualquer forma, o fato de o ambiente em questão representar um desafio para o sistema baseado no SIFT, não invalida as avaliações feitas. Para que se decida de maneira definitiva sobre a usabilidade das bases de dados obtidas com cada um destes ângulos de amostragem, é necessário utilizá-las em aplicação de reconhecimento.

Se no caso da avaliação da amostragem, o baixo número de pontos-chave por modelo dificulta a avaliação, no caso da criação de ligações ela é benéfica. A existência de poucos pontos por modelo dificulta a criação das ligações, o que tende a privilegiar a técnica mais robusta. Com os resultados da avaliação da amostragem, decidiu-se avaliar a criação de ligações, somente para os ângulos de 10° e 15° graus, uma vez que estas geram a maior quantidade de modelos a serem ligados. Como a aleatoriedade observada foi baixa, apresenta-se os resultados relativos a apenas um teste executado com cada

ângulo de amostragem.

Tabela 9 – Resultados dos testes de construção de ligações. Os valores se referem apenas aqueles obtidos pela inversão e recálculo, sem somar as ligações obtidas durante o processamento inicial.

		Inversão	**Recálculo**	
10°	Ligações entre Modelos	402	395	-1.7%
	Ligações	9931	9745	-1.9%
	Desvio padrão ligações ente modelos	1.829	1.227	
	Desvio padrão ligações	59.939	43.263	
15°	Ligações para Modelos	709	698	-1.6%
	Ligações criadas	17318	17187	-0.8%
	Desvio padrão ligações ente modelos	1.972	1.193	
	Desvio padrão ligações	61.889	45.353	

A partir dos testes, apresentados na tabela 9 verificou-se que o método de recálculo cria uma quantidade ligeiramente inferior de ligações entre pontos-chave e entre modelos, o que indica equivalência quanto à quantidade de informação adicionada. Porém, percebe-se que o método de recálculo consegue espalhar melhor as ligações entre as poses. Para ambos os ângulos de amostragem, verificou-se que o desvio-padrão da distribuição de ligações entre modelos e entre pontos-chave é maior para o método de inversão. Como há um número fixo de ligações, o valor mais alto do desvio padrão indica que estas se acumulam em maior quantidade em um número menor de modelos.

Para verificar esta situação decidiu-se comparar o número de modelos de pose para os quais cada método gera mais ligações. Desta forma, se um método concentra as ligações em um único modelo, então ela só terá resultado mais alto para este modelo específico. O método que distribui melhor as ligações terá mais modelos nos quais ele terá criado a maior quantidade de ligações. Na tabela 10 é possível verificar o resultado desta análise.

Olhando a tabela, verifica-se que o método de recálculo atribui ligações a um maior número de modelos, em ambos os ângulos de amostragem. No caso das base de dados testadas, o método de recálculo produz um número maior ou igual de ligações em pelo menos 66% dos modelos de pose. Novamente, a avaliação final sobre a escolha dos métodos depende da implementação do sistema de reconhecimento de objetos, para testar as duas

Tabela 10 – Contagem dos modelos para os quais cada método atribuiu mais ligações

	15°	10°
	Contagem (%)	Contagem (%)
Recálculo	96 (40.2)	166 (46.6)
Inversão	76 (31.8)	121 (34.0)
Iguais	67 (28.0)	69 (19.4)
Total de Modelos	239	356

possibilidades.

Por fim, conclui-se que através das implementações feitas durante esta pesquisa de mestrado, foi possível testar a aplicação de métodos de agrupamento tridimensional de poses, para criação de bases de dados para sistemas de reconhecimento de objetos baseados no SIFT. Conseguiu-se gerar várias configurações para geração de bases de dados, e percebeu-se que o uso de ângulos de amostragem superiores a 30°, tem dificuldade em gerar ligações para que se possa agrupar as poses, e que o recálculo das ligações permite que estes sejam melhor distribuídos entre os modelos. A partir destes resultados, pode-se considerar que o objetivo 2 foi atingido, concluindo o conjunto de objetivos desta pesquisa de mestrado e este capítulo de resultados. No próximo, serão apresentadas as conclusões e as sugestões de trabalhos futuros.

6 CONCLUSÃO

A criação de um *framework* para construção de simuladores de ambientes tridimensionais e as bases para a implementação de um sistema de reconhecimento de objetos para ser aplicado à robótica móvel foram as principais contribuições deste trabalho.

A primeira contribuição surgiu do estudo das ferramentas disponíveis e da verificação de suas funcionalidades. A avaliação destas foi feita através de adaptação da metodologia de Kramer e Scheutz (2007). A partir das análises percebeu-se que as ferramentas testadas (MORSE e Gazebo), não forneciam por padrão, os recursos necessários para suprir o critério de qualidade gráfica. Estudando a documentação do motor de renderização OGRE, utilizado pelo Gazebo, conseguiu-se encontrar soluções para os problemas gráficos.

Como o estudo foi feito principalmente através de implementação de tutoriais, conseguiu-se por meio destes, as condições necessárias para a simulação desejada. Percebeu-se que seria mais complicado transportar as soluções para as ferramentas previamente avaliadas, do que criar a simulação sobre a estrutura de código dos tutoriais. Desta forma criou-se um *framework* para o desenvolvimento de simulações de aplicações de visão computacional baseado nos códigos elaborados durante o estudo dos tutoriais do renderizador. Como visto no capítulo 5, este *framework* permitiu a criação de simulações de um ambiente de escritório com significativa fidelidade, sendo possível emular neste ambiente, um robô com câmeras estéreo. Através de um teste de calibração pode-se confirmar a exatidão disponibilizada pelo sistema.

Considerando as capacidades atuais do simulador, e os resultados da geração de bases de dados, seria interessante que trabalhos futuros incrementassem as capacidades gráficas, através do uso de soluções de computação gráfica mais completas. Sugere-se a adição de técnicas baseadas em mapas de textura para melhorar a aparência dos objetos. A melhoria da aparência seria de fundamental importância para aumentar o grau de realismo do simulador, o que aumentaria a qualidade dos testes realizados por meio deste. Já se possui uma implementação da técnica de iluminação *ambient occlusion*, porém a falta de conhecimento em design gráfico do autor impediu a criação de texturas para o uso desta.

Se possível, seria interessante utilizar métodos de iluminação ainda mais avançados. Um exemplo é a técnica *global illumination*[1] que modela luzes indiretas. Para melhorar o desempenho temporal pode-se modificar as malhas poligonais usadas no ambiente. As malhas poligonais utilizadas atu-

[1] http://www.advancedglobalillumination.com/

almente tem origem em base de dados de arquitetura, e por isso são muito complexas para simulação *online*. Também nessa área já existe implementação de técnicas mais adequadas (*normal mapping*), mas a falta de conhecimentos na área gráfica por parte do acadêmico impediu o seu uso. Outro possível interesse seria verificar como correlacionar os resultados obtidos no simulador com aqueles obtidos na realidade.

A realização do trabalho se mostrou mais complicada na parte relativa ao sistema de reconhecimento de objetos. Além da natureza complexa do problema, a elevada quantidade de abordagens possíveis se mostrou um complicador. Nas fases iniciais dos estudos, pensou-se em usar uma abordagem derivada da área de reconhecimento de padrões. Foram realizados análises de algumas soluções deste campo (Grauman e Darrell (2005) e Lazebnik et al. (2006) por exemplo), os quais usam os SIFT em conjunto com técnicas de aprendizado de máquinas, como o SVM (*Support Vector Machine*). Esta abordagem foi descartada porque as soluções desta área apenas indicam a existência do objeto na imagem, e não sua posição.

Como esta é a informação mais importante para que o robô possa se movimentar, percebeu-se que seria necessário desenvolver um sistema muito parecido com o proposto por Lowe (2004), para encontrar a posição do objeto. Um sistema de reconhecimento utilizando as duas abordagens, poderia usar o reconhecimento de padrões para identificar as classes dos objetos na imagem, com o posterior uso do sistema de Lowe para refinamento e definição da posição. Levando isso em consideração, deixou-se a execução desta proposta para um trabalho futuro.

Em relação a pesquisa desenvolvida para geração de bases de dados, sem dúvida a maior contribuição foi na proposição de uma modificação na maneira de criar as ligações definidas pelo algoritmo de agrupamento tridimensional de poses. O método de recálculo de ligações se mostrou mais equilibrado, distribuindo melhor entre os modelos de pose as ligações criadas. Para avaliação da utilidade prática deste método é necessário mais pesquisas, sendo de primordial importância o complemento da presente pesquisa, completando-se o sistema de reconhecimento de objetos baseado no SIFT.

A partir do que já foi desenvolvido nesta pesquisa de mestrado, ainda é necessário implementar a parte probabilística do sistema e, a partir disto, testar novamente as abordagens de criação de base de dados para obter a decisão final sobre o desempenho das mesmas.

Outro trabalho futuro sugerido é a implementação de técnicas de mapeamento e localização simultânea (SLAM) baseados nas mesmas características usadas para o reconhecimento de objetos. Isso permitiria a execução do processamento visual em dois fluxos de processamento. Um para tratar de forma mais reativa os problemas de movimentação, seria feito através da im-

plementação SLAM, e teria que ser rápido. O segundo seria desempenhado pelo algoritmo de reconhecimento de objetos, e cuidaria da parte cognitiva de visão, podendo ser mais lento. Miro et al. (2006) e Se et al. (2002) são duas referências para a elaboração do SLAM.

Sugere-se também uma pesquisa com âmbito mais teórico, através de um estudo mais aprofundado do SIFT para sugestão de modificações. Durante os estudos feitos na presente dissertação, percebeu-se que seriam necessárias informações de forma para melhorar os resultados, principalmente para soluções da correspondência de características obtidas em texturas com padrões fortes. Bosch et al. (2007) apresentaram um exemplo de uma estratégia destas em seu descritor o PHOG (*Pyramid of Histograms of Orientation Gradients*). Ele é baseado no SIFT para descrição de aparência e um descritor de forma baseado em histogramas de orientação de bordas obtidas com o detector de *Canny*.

Concluindo, pode-se perceber que há vários trabalhos a serem desenvolvidos em continuação deste, o que comprova sua importância como possível desencadeador de novas pesquisas.

REFERÊNCIAS BIBLIOGRÁFICAS

AYACHE, N.; LUSTMAN, F. Trinocular stereovision for robotics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 13, n. 1, p. 73–85, 1991.

BALLARD, D. H. Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, v. 11, n. 2, p. 111–122, 1981.

BONIN-FONT, F.; ORTIZ, A.; OLIVER, G. Visual Navigation for Mobile Robots: A Survey. *Journal of Intelligent and Robotic Systems*, v. 53, n. 3, p. 263–296, 2008. ISSN 0921-0296.

BOSCH, A.; ZISSERMAN, A.; MUNOZ, X. Representing shape with a spatial pyramid kernel. In: *Proceedings of the 6th ACM international conference on Image and video retrieval*. New York, NY, USA: ACM, 2007. (CIVR '07), p. 401–408. ISBN 978-1-59593-733-9.

CASTILLO-PIZARRO, P.; ARREDONDO, T. V.; TORRES-TORRITI, M. Introductory Survey to Open-Source Mobile Robot Simulation Software. In: *Latin American Robotics Symposium and Intelligent Robotics Meeting*. Sao Bernardo do Campo, Brazil: IEEE Computer Society Press, 2010. p. 150–155. ISBN 978-1-4244-8639-7.

CHEN, D.-Y.; TIAN, X.-P.; SHEN, Y.-T.; OUHYOUNG, M. On visual similarity based 3D model retrieval. *Computer graphics*, v. 22, n. 3, p. 223–232, set. 2003. ISSN 0167-7055.

CHEN, K.; TSAI, W. Vision-based autonomous vehicle guidance for indoor security patrolling by a SIFT-based vehicle-localization technique. *IEEE Transactions on Vehicular Technology*, v. 59, n. 7, p. 3261–3271, 2010.

CHOSET, H.; LYNCH, K. M.; HUTCHINSON, S.; KANTOR, G. A.; BURGARD, W.; KAVRAKI, L. E.; THRUN, S. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 2005. 603 p. ISBN 9780262033275.

DESOUZA, G. N.; KAK, A. C. Vision for mobile robot navigation: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 2, p. 237–267, 2002. ISSN 01628828.

DIXON, J.; HENLICH, O. Mobile robot navigation. *Information Systems Engineering Year, Imperial College*, v. 2, p. 1–10, 1997. <http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/jmd/>.

ECHEVERRIA, G.; LASSABE, N.; DEGROOTE, A.; LEMAIGNAN, S. Modular open robots simulation engine: MORSE. *International Conference on Robotics and Automation*, IEEE Computer Society Press, p. 46–51, maio 2011.

GOSHTASBY, A. A. *2-D and 3-D Image Registration: For Medical, Remote Sensing, and Industrial Applications*. Hoboken, NJ: Wiley-Interscience, 2005. ISBN 0471649546.

GRAUMAN, K.; DARRELL, T. The pyramid match kernel: Discriminative classification with sets of image features. In: *International Conference on Computer Vision*. Beijing: IEEE Computer Society, 2005.

HARRIS, A.; CONRAD, J. M. Survey of popular robotics simulators, frameworks, and toolkits. In: *Southeastcon, 2011 Proceedings of IEEE*. Nashville, TN: IEEE Computer Society Press, 2011. p. 243–249. ISBN 978-1-61284-739-9.

HARTLEY, R.; ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press, 2003. (Cambridge books online). ISBN 9781139449144.

KIM, D.; NEVATIA, R. Symbolic navigation with a generic map. *Autonomous Robots*, v. 88, n. 6, p. 69–88, 1999.

KLIPPENSTEIN, J.; ZHANG, H. Quantitative evaluation of feature extractors for visual slam. In: *Fourth Canadian Conference on Computer and Robot Vision*. Montreal,Que: IEEE Computer Society Press, 2007. p. 157–164.

KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. *International Conference on Intelligent Robots and Systems*, IEEE Computer Society Press, v. 3, p. 2149–2154, 2004.

KRAMER, J.; SCHEUTZ, M. Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, v. 22, n. 2, p. 101–132, dez. 2007. ISSN 0929-5593.

LAZEBNIK, S.; SCHMID, C.; PONCE, J. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In: *Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE Computer Society, 2006. v. 2, p. 2169–2178. ISBN 0-7695-2597-0.

LINDEBERG, T. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, p. 224–270, 1994.

LOWE, D. Object recognition from local scale-invariant features. In: *7th International Conference on Computer Vision*. [S.l.]: IEEE Computer Society Press, 1999. v. 2, p. 1150–1157.

LOWE, D. Local feature view clustering for 3D object recognition. In: *Conference on Computer Vision and Pattern Recognition.* Kauai, HI: IEEE Computer Society, 2001. v. 1, p. I–682–I–688. ISBN 0-7695-1272-0.

LOWE, D. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, v. 60, n. 2, p. 91–110, nov. 2004. ISSN 0920-5691.

MADHAVAN, R.; DURRANT-WHYTE, H.; DISSANAYAKE, G. Natural landmark-based autonomous navigation using curvature scale space. In: *IEEE International Conference on Robotics and Automation*. Washington, DC: IEEE Computer Society Press, 2002. v. 4, n. May 2002, p. 3936–3941. ISBN 0-7803-7272-7.

MAIMONE, M.; CHENG, Y.; MATTHIES, L. Two years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, v. 24, n. 3, p. 169–186, mar. 2007. ISSN 15564959.

MEYER, J.; SENDOBRY, A.; KOHLBRECHER, S.; KLINGAUF, U.; STRYK, O. Comprehensive simulation of quadrotor uavs using ros and gazebo. In: NODA, I.; ANDO, N.; BRUGALI, D.; KUFFNER, J. (Ed.). *Simulation, Modeling, and Programming for Autonomous Robots*. Berlin, Germany: Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7628). p. 400–411. ISBN 978-3-642-34326-1.

MIKOLAJCZYK, K. *Detection of local features invariant to affine transformations*. 171 p. Tese (Phd. Thesis) — Institut National Polytechnique de Grenoble,France, 2002.

MIKOLAJCZYK, K.; SCHMID, C. Performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, v. 27, n. 10, p. 1615–30, out. 2005. ISSN 0162-8828.

MIRO, J.; ZHOU, W.; DISSANAYAKE, G. Towards Vision Based Navigation in Large Indoor Environments. In: *International Conference on Intelligent Robots and Systems*. Beijing, China: IEEE Computer Society Press, 2006. p. 2096–2102. ISBN 1-4244-0258-1.

MOREL, J. ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, v. 2, n. 2, 2009.

NEBOT, E. Sensors Used for Autonomous Navigation. In: TZAFESTAS, S. (Ed.). *Advances in Intelligent Autonomous Systems*. Dordrecht, NL: Springer Netherlands, 1999, (International Series on Microprocessor-Based and Intelligent Systems Engineering, v. 18). p. 135–156. ISBN 978-94-010-6012-7.

NISTER, D.; NARODITSKY, O.; BERGEN, J. Visual odometry. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Washington, DC: IEEE Computer Society Press, 2004. v. 1, p. 652–659. ISBN 0-7695-2158-4.

NIXON, M. S.; AGUADO, A. S. *Feature Extraction & Image Processing*. London, UK: Academic Press, 2008. 406 p. ISBN 9780123725387.

PINAGE, F.; CARVALHO, J. R. H.; NETO, J. P. d. Q. Natural Landmark Tracking Method to Support UAV Navigation over Rain Forest Areas. In: *2012 Brazilian Symposium on Computing System Engineering*. Natal,RN: IEEE, 2012. p. 105–110. ISBN 978-0-7695-4929-3.

RAMISA, A.; ALDAVERT, D.; VASUDEVAN, S.; TOLEDO, D.; MANTARAS, R. L. de. Evaluation of the sift object recognition method in mobile robots. In: *Proceedings of the 2009 conference on Artificial Intelligence Research and Development: Proceedings of the 12th International Conference of the Catalan Association for Artificial Intelligence*. Amsterdam, The Netherlands: IOS Press, 2009. p. 9–18.

SANTOS-VICTOR, J.; SANDINI, G.; CUROTTO, F.; GARIBALDI, S. Divergent stereo for robot navigation: learning from bees. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: IEEE Computer Society Press, 1993. p. 434–439. ISBN 0-8186-3880-X.

SE, S.; LOWE, D.; LITTLE, J. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *The International Journal of Robotics Research*, v. 21, n. 8, p. 735–758, ago. 2002. ISSN 0278-3649.

STIVANELLO, M. *Desenvolvimento de uma biblioteca para sistemas de visão estereoscópia para robótica móvel*. Florianópolis, Brasil: Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia Elétrica., 2008.