Uma abordagem colaborativa bioinspirada para localização e mapeamento simultâneos de agentes móveis utilizando visão monocular

Evandro Luís Souza Falleiros

Uma abordagem colaborativa bioinspirada para localização e mapeamento simultâneos de agentes móveis utilizando visão monocular

Evandro Luís Souza Falleiros

Orientador: *Prof. Dr. Renato Porfirio Ishii*

Dissertação apresentada à Faculdade de Computação da Universidade Federal de Mato Grosso do Sul como parte dos requisitos necessários à obtenção do título em Mestre em Ciência de Computação.

UFMS - Campo Grande
Dezembro - 2014

Aos meus pais,
Márcio Luís Falleiros e Célia Regina Souza Falleiros,

Ao meu irmão,
Flávio Luís Souza Falleiros,

À minha esposa,
Nátalli Macedo Rodrigues Falleiros,

Ao meu filho,
Gustavo Rodrigues Falleiros,

Ao meu orientador,
Prof. Dr. Renato Porfirio Ishii.

À minha família e aos sinceros e bons amigos.

Agradecimentos

Primeiramente, agradeço aos meus pais, Márcio Luís Falleiros e Célia Regina Souza Falleiros, por serem pais presentes e amorosos. Presentearam-me, durante a vida, com uma educação exemplar, acima de tudo. Foram, são e sempre serão minha fonte de inspiração. Nunca economizaram esforços para que eu pudesse chegar aqui. Agradeço, de coração, por tudo que sempre me proporcionaram. Também agradeço imensamente ao meu irmão, Flávio Luís Souza Falleiros, que sempre esteve ao meu lado nos momentos que precisei. Obrigado por ser um irmão verdadeiro e companheiro para todas as horas. O seu apoio foi essencial para que eu seguisse meu caminho.

Agradeço à minha esposa amada, Nátalli Macedo Rodrigues Falleiros, que acompanhou cada um dos passos que dei em direção à essa conquista. Suportou, firme, as difíceis viagens e o tempo que fiquei fora de casa cursando as disciplinas. Apoiou-me em todos os momentos, sem medir esforços. Mesmo nas horas mais difíceis foi capaz de me manter em pé e lutando. Nas horas de desespero, soube me acalmar. Acompanhou-me, sabiamente, em cada uma das angústias e dúvidas que surgiram. Agradeço, imensamente, o apoio que me deste nesse processo. Você é parte integrante e essencial nesta conquista. E, acima de tudo, você sempre será um exemplo para mim.

Juntamente com Nátalli, agradeço a existência de nosso filho Gustavo Rodrigues Falleiros, que nasceu no decorrer do Mestrado. Meu filho amado, é por você que luto! O seu sorriso, o seu aconchego e o seu jeitinho carinhoso me motivam a continuar. A vida faz todo o sentido com você por perto.

Agradeço também à minha Sogra, Maria de Fátima Macedo Alves, que foi compreensiva e companheira nos momentos mais difíceis. Obrigado por tudo! Agradeço a meus cunhados e cunhadas pela paciência e compreensão. Mesmo

estando longe, cada um de vocês faz parte desta conquista.

Agradeço imensamente à Graziela Rabelo Marquez e à Gabriela Rabelo por me receberem de braços abertos em vossa residência no período que viajei semanalmente para Campo Grande para cumprir as disciplinas do Mestrado. Sou grato e sempre me lembrarei de vocês com muito carinho. Essa conquista também é de vocês.

Agradeço ao meu Orientador, Prof. Dr. Renato Porfírio Ishii, pela imensa paciência e profissionalismo. Você me recebeu de braços abertos sem nem mesmo conhecer meu trabalho. Agradeço, imensamente, o seu apoio. Não sabes, mas durante esse período de orientação, aprendi muito contigo. Tornastes fonte de inspiração em minha profissão docente. Vou levar seus ensinamentos para toda vida!

Agradeço ao Prof. Dr. Rodrigo Calvo. Seu apoio e conhecimento foram essenciais para que esta dissertação fosse concluída. Agradeço por ter me atendido e auxiliado sem nem mesmo me conhecer. Saiba que sua Tese me inspirou e sou grato por ter você compartilhando o seu conhecimento comigo.

Agradeço também ao Prof. Dr. Edson Takashi Matsubara. Suas aulas de Inteligência Artificial e cursos sobre robótica móvel no programa DESTACOM fizeram com que eu me apaixonasse pela área de IA. Obrigado pelo conhecimento compartilhado. Em seu nome, agradeço ao Programa de Mestrado da FACOM-UFMS. Sou grato pelos momentos de conhecimento partilhados.

Agradeço, por fim, a todos os bons amigos e companheiros de trabalho. São muitos que não posso nomeá-los. Obrigado a todos pelo apoio e compreensão.

Resumo

Aplicações com múltiplos robôs vem sendo discutidas exaustivamente nos últimos anos e são consideradas problemas fundamentais na robótica móvel. Entretanto, o desenvolvimento de aplicações em tempo-real com múltiplos robôs é geralmente uma tarefa difícil, uma vez que existe a necessidade de se construir um ambiente robusto que suporta tal cenário de implementação. Esta dissertação propõe um sistema bio-inspirado, denominado PheroSLAM, que adota uma versão estendida da abordagem Colônia de Formigas para coordenar múltiplos robôs em tarefas de exploração e vigilância de ambientes. No PheroSLAM, cada robô deposita feromônio artificial repulsivo ao seu redor, criando uma trilha repulsiva. Essa trilha deve ser evitada pelos demais robôs, uma vez que denotam áreas que foram recentemente visitadas. Um algoritmo SLAM baseado em visão, denominado MonoSLAM, também é utilizado para prover informações de odometria visual para múltiplos robôs. O MonoSLAM constrói mapas tridimensionais baseados em *features*, considerando que cada robô deve ser capaz de se localizar no ambiente explorado. O sistema proposto é uma contribuição relevante considerando aplicações para múltiplos robôs, como a construção de mapas ou a exploração e vigilância de ambientes. Evidências empíricas mostraram que o PheroSLAM apresenta boa dispersibilidade, o que promove o aumento da cobertura de ambientes. Os resultados experimentais mostraram que a estratégia de coordenação é eficiente e satisfatória para o cumprimento de tarefas de exploração e vigilância.

Abstract

Multi-robot applications have been extensively discussed in the past years and they are considered a fundamental problem in mobile robots. Nevertheless, to develop multi-robot real-time applications is usually a complex task, since there is a need of to construct robust development environments that support this implementation scenario. This dissertation proposes a bio-inspired multi-robot system, named PheroSLAM, that adopt an extended version of Ant Colony approach to coordinate multiple-robot teams in exploration and surveillance tasks. In PheroSLAM, every robot deposits repulsive artificial pheromone around itself, creating a repulsive trail. This trail must be avoid by the other robots, since it denotes an area that have been recently explored. A vision-based SLAM, named MonoSLAM, is also used to provide visual odometry information to multi-robot teams. MonoSLAM builds a 3D feature-based map, considering that each robot must be able to localize itself within the explored environment. The proposed system is a relevant contribution considering multi-robot applications, such as map construction or collaborative environment exploration and surveillance tasks. Empirical evidence showed that PheroSLAM has good dispersibility, which promoted increased coverage in an environment. The experimental results showed that the coordination strategy is efficient and satisfactory to accomplish exploration and surveillance tasks.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ACO Ant Colony Optimization

IAS-SS Inverse Ant System-Based Surveillance System

SLAM Simultaneous Localization and Mapping

SMR Sistemas Multi-Robôs

Lista de Algoritmos

Sumário

CAPÍTULO

1

Introdução

1.1 Contextualização

Nos últimos anos, robôs têm sido amplamente utilizados para a resolução de diversos problemas. Tarefas específicas, como a construção de mapas tridimensionais (Markham et al., 2007), a busca por objetos diversos em um ambiente desconhecido (Shubina e Tsotsos, 2010), a detecção de odores em ambientes inóspitos (Turduev et al., 2012), a gestão de desastres (Chaudhury, 2011) e a inspeção industrial (Schmidt e Berns, 2013) são exemplos nos quais um ou mais robôs podem ser utilizados a fim de resolver os problemas identificados.

Grande parte desses problemas podem, inevitavelmente, colocar a vida humana em risco, uma vez que os ambientes que serão explorados podem ser inóspitos. Nesse sentido, considerando tarefas que não devem ser realizadas por seres humanos, é razoável que robôs móveis sejam empregados para resolver tais problemas. Em alguns casos, inclusive, faz mais sentido utilizar uma equipe de robôs, que poderá resolver problemas utilizando **estratégias autônomas de colaboração**.

Os conceitos de autonomia e colaboração consistem basicamente na execução de tarefas de forma autônoma a partir da aplicação de técnicas de colaboração entre os agentes envolvidos. Os agentes envolvidos devem executar

tarefas com pouca ou nenhuma interação humana. Para tal, a tomada de decisões para o cumprimento de uma tarefa é responsabilidade inerente aos próprios agentes.

Estratégias de colaboração são definidas para que todos os agentes envolvidos colaborem e interajam entre si, a fim de cumprir uma tarefa com maior dinamicidade, rapidez e corretude. Segundo Franklin e Graesser (1997), um agente autônomo interage com um ambiente por meio de sensores, que proveem dados de entrada relativos à percepção do ambiente, e por meio de atuadores, que permitem que esse agente atue no ambiente explorado.

Diversos sensores são utilizados para a percepção do ambiente. Os sensores mais utilizados são: sensores odométricos, GPS (Global Positioning System - sistema de posicionamento global), *laser scanners*, sensores ópticos unidimensionais, sensores a *laser* bidimensionais, sensores a *laser* tridimensionais de alta definição, sensores sonares bidimensionais ou tridimensionais, câmeras estereoscópicas, dentre outros. Esses sensores são geralmente utilizados em conjunto, a fim de aumentar a precisão na localização e no mapeamento dos ambientes explorados.

Contudo, a utilização de sensores como GPS e *laser scanners* pode acarretar alguns problemas. Segundo Royer et al. (2007), sensores GPS não são adequados para ambientes fechados. Dessa forma, sugere-se que outros sensores sejam utilizados quando o ambiente a ser explorado não for um ambiente externo. Por outro lado, os sensores *laser scanners* costumam gerar grandes volumes de dados e são sensores com custo elevado.

Considerando tarefas nas quais agentes autônomos devem interagir entre si e com o ambiente explorado, um importante problema a ser resolvido é identificar a posição e a orientação de um agente. Cada agente em uma equipe de colaboração precisa saber sua própria localização com relação aos demais que integram sua equipe. Para Aulinas et al. (2008), essa habilidade é essencial, considerando tarefas de vigilância e exploração de ambientes.

Além disso, o sensoriamento do ambiente deve ser efetivo, a fim de construir um mapa representativo do ambiente explorado. Este mapa é essencial para que os agentes envolvidos possam se localizar no ambiente. Em alguns casos, tais agentes não possuem informações prévias sobre o ambiente que está sendo explorado e, consequentemente, não existe um mapa que o represente. Nesses casos, um mapa deve ser construído ao mesmo tempo que os agentes tentam se localizar no ambiente.

O problema de localização e mapeamento simultâneos é conhecido como *Simultaneous Localization and Mapping* (SLAM) e consiste na tarefa de conciliar a localização e o mapeamento de um ambiente em tempo de navegação, de forma que os resultados obtidos sejam condizentes com a realidade. Assim, tendo como base a definição para agentes autônomos de Franklin e Graesser (1997), entende-se que a resolução de problemas que envolvem SLAM é essencial para a construção de cenários cujos agentes possam realizar uma ou mais tarefas específicas de forma autônoma e colaborativa, sem a intervenção humana.

Entretanto, um agente deve conhecer um objetivo inicial para que possa navegar. Franklin e Graesser (1997) afirmam que um agente age através de objetivos e alvos que são impostos. Para que uma ação qualquer seja executada, o agente deve conhecer uma representação do modelo de ambiente explorado, para que possa, consequentemente, se localizar neste ambiente. Nessa representação, pode haver a definição de um mapa referente ao ambiente e a definição dos objetivos que devem ser atingidos. Estes objetivos possuem medidas de desempenho ou funções de utilidade que expressam as preferências do agente.

Tratando-se de ambientes desconhecidos ou parcialmente conhecidos, os agentes envolvidos tendem a enfrentar problemas relativos à sua localização e à construção incremental de um mapa, no qual são armazenados os locais já explorados e a sua estimativa de localização dentro do ambiente. A tarefa de um agente ter conhecimento de sua própria localização em um instante de tempo e, simultaneamente, construir um mapa do ambiente desconhecido é complexa, uma vez que o sensoriamento e a percepção do ambiente demandam precisão.

Infelizmente, tais atividades são comprometidas, frequentemente, pela imprecisão dos sensores utilizados nos robôs, bem como pelas irregularidades existentes no terreno explorado. Vale ressaltar que a localização de cada um dos agentes envolvidos está fortemente ligada à construção ou conhecimento de um mapa do ambiente e, em diversos escopos, deseja-se que o posicionamento destes agentes seja sempre o mais preciso possível.

Trabalhos recentes apresentados por Smith et al. (1990); Cho e Kim (2010); Myung et al. (2010); Murphy (2010); Montemerlo et al. (2002a, 2003); Brekke e Chitre (2013); Zhang et al. (2013); Carlone et al. (2014); Lee et al. (2013); Sim et al. (2005); Newcombe et al. (2011); Steder et al. (2008); Wang e Chen (2013);

Zou (2013) tem proposto algoritmos para resolver e melhorar o problema de SLAM, considerando diferentes tipos de sensores para adquirir dados com erros estatisticamente independentes.

Royer et al. (2007) complementa afirmando que, para obter resultados mais confiáveis em uma aplicação de escala real, diversos outros sensores devem ser utilizados em conjunto a fim de aumentar a confiabilidade dos resultados gerados. Em tarefas que envolvem a reconstrução tridimensional de um ambiente desconhecido, por exemplo, pode-se afirmar que a argumentação de Royer et al. (2007) é coerente, uma vez que se espera um mapa 3D condizente com a realidade.

Porém, em alguns casos específicos, não há a necessidade de que o mapa gerado para o ambiente explorado seja perfeito. Espera-se, contudo, que o objeto de busca seja encontrado e eventualmente resgatado. Concomitantemente, identifica-se uma quantidade considerável de aplicações e problemas que envolvem diversas estratégias para a exploração, vigilância e construção de mapas.

Visando a diminuição dos custos inerentes ao sensores supracitados, Royer et al. (2007) afirma que a utilização de um sistema de localização baseado em visão é uma boa maneira de reduzir os custos e o tamanho físico dos agentes utilizados. Royer et al. (2007) ainda acrescenta que a visão é um interessante sensor, uma vez que grande parte dos ambientes considerados apresentam diversas características que podem ser facilmente identificadas. Além disso, o campo de visão é utilizado por grande parte dos animais como mecanismo de localização espacial.

Atualmente, observa-se que a visão computacional vem contribuindo com uma vasta gama de soluções inerentes à percepção e detecção de objetos, sendo amplamente utilizada na robótica com cenários e ambientes distintos. Uma das soluções que encontra-se em frequente crescimento é a resolução de problemas que envolvem a localização e o mapeamento simultâneos utilizando o campo de visão pura. Segundo Karlsson et al. (2005), desde 2005, há uma pesquisa intensa sobre SLAM visual (VSLAM) utilizando sensores primariamente visuais (câmeras), devido à crescente onipresença das câmeras em dispositivos móveis.

O presente trabalho propõe um sistema SLAM bio-inspirado, denominado PheroSLAM, que utiliza o campo de visão pura como único sensor para a obtenção de dados odométricos. O PheroSLAM é baseado em uma versão modi-

fica da abordagem Colônia de Formigas proposta por Dorigo (1992). Adotou-se como abordagem para a obtenção de odometria visual o algoritmo MonoSLAM proposto por Davison et al. (2007).

Segundo Campbell et al. (2005), o termo odometria visual remete à uma estimativa incremental sobre a movimentação de um robô, considerando uma sequência de imagens de vídeo obtida de uma câmera acoplada a este. Além de prover dados para odometria visual, o algoritmo MonoSLAM é utilizado para construir e manter um mapa tridimensional probabilístico de características salientes (*features*) identificadas no ambiente explorado.

Na abordagem PheroSLAM, cada robô é capaz de estimar sua posição no ambiente explorado através do algoritmo MonoSLAM. Entretanto, uma estratégia eficiente de exploração autônoma é necessária para que cada robô possa se locomover no ambiente explorado. Dessa forma, propõe-se uma versão modifica da abordagem de Colônia de Formigas como extensão da abordagem *Inverse Ant System-Based Surveillance System* (IAS-SS) apresentada por Calvo et al. (2011).

A abordagem IAS-SS utiliza feromônio como substância repulsiva para coordenar múltiplos robôs em tarefas de exploração e vigilância de ambientes. Enquanto se movem no ambiente em busca de um objetivo específico, cada robô apresenta um comportamento colaborativo repulsivo à substância (feromônio) depositada no ambiente.

O comportamento repulsivo observado no IAS-SS é inverso ao comportamento atrativo tradicional observado na abordagem de Colônia de Formigas. No IAS-SS, a repulsão às substâncias depositadas tem como objetivo previnir que áreas recém visitadas em um ambiente sejam visitadas novamente por outros robôs em um curto espaço de tempo.

Baseado no comportamento repulsivo do IAS-SS, o PheroSLAM propõe a exploração autônoma de ambientes através do algoritmo *Low Pheromone Concentration Search* (LPCS). O algoritmo LPCS consiste na busca por áreas que apresentam as menores concentrações de feromônio ao redor de um robô. Na abordagem proposta, definiu-se que tais áreas apresentam o maior ganho de informação em uma iteração, uma vez que são áreas possivelmente não exploradas ou não recentemente exploradas por outros robôs.

Durante a execução do PheroSLAM, a cada iteração de exploração, o algoritmo LPCS fornece a um robô uma direção alvo de exploração. Tal direção deve ser mantida até que uma nova direção seja provida pelo algoritmo. Con-

tinuamente, o algoritmo LPCS busca por novas regiões a serem exploradas. A ideia principal é que o algoritmo LPCS seja capaz de guiar cada robô a áreas que não foram visitadas anteriormente.

Para fins de validação da proposta, foram realizados experimentos práticos em ambientes controlados. Os resultados experimentais mostram que a abordagem proposta é suficientemente robusta para suportar aplicações multi-robôs em tarefas de exploração e vigilância de ambientes internos desconhecidos.

A seguir, os objetivos e as contribuições do presente trabalho são apresentados e discutidos.

1.2 Objetivos e contribuições

Essa dissertação tem como objetivo principal o desenvolvimento de um sistema bio-inspirado que utiliza agentes autônomos para a resolução de problemas de localização e mapeamento simultâneos de forma colaborativa utilizando apenas o campo de visão pura como sensor para a obtenção de dados de odometria. Tem-se como objetivos específicos:

1. Possibilitar a navegação e a exploração de ambientes desconhecidos utilizando estratégias de colaboração bio-inspiradas;
2. Explorar abordagens para a obtenção de odometria visual a partir de câmeras monoculares;
3. Estender o sistema MonoSLAM desenvolvido por Davison et al. (2007), para que a exploração de ambientes seja realizada de forma autônoma e colaborativa.
4. Verificar, por meio de experimentos em ambientes reais e simulados, a robustez e a eficiência dos algoritmos propostos.

1.3 Organização do trabalho

Esta dissertação de mestrado está organizada nos seguintes capítulos:

- Capítulo 2: são apresentadas definições diversas sobre os temas abordados nesta dissertação, com enfoque em: localização e mapeamento

simultâneos, odometria visual, abordagens colaborativas para sistemas com múltiplos robôs, abordagens bio-inspiradas para exploração e vigilância de ambientes desconhecidos;

- Capítulo 3: é apresentado um algoritmo de localização e mapeamento simultâneos que utiliza o campo de visão pura para prover dados de odometria visual;

- Capítulo 4: apresenta-se, neste capítulo, abordagens de inteligência coletiva baseadas no comportamento de formigas. Busca-se contextualizar duas frentes de pesquisa que são diretamente abordadas nesta dissertação, sendo estas: otimização por colônia de formigas e sistema de vigilância baseada na modificação do sistema de colônia de formigas;

- Capítulo 5: é descrita a abordagem de inteligência coletiva proposta nesta dissertação, aplicada a problemas de exploração e vigilância de ambientes desconhecidos. Após as definições formais da abordagem proposta, o sistema desenvolvido é apresentado e discutido. Por fim, os resultados experimentais são apresentados e discutidos;

- Capítulo 6: apresenta-se a conclusão desta dissertação, as devidas contribuições, as limitações identificadas e as propostas para trabalhos futuros.

CAPÍTULO

2

Conceitos e trabalhos relacionados

2.1 Considerações iniciais

Neste capítulo, são apresentadas algumas soluções propostas na literatura para os seguintes problemas: localização e mapeamento simultâneos, mapeamento de ambientes, robótica colaborativa e estratégias de colaboração e visão monocular em problemas de SLAM. Os trabalhos relacionados, bem como os conceitos envolvidos em tais soluções, são apresentados e discutidos a seguir.

2.2 Localização e mapeamento simultâneos

Estratégias para a navegação de robôs são guiadas pelo conhecimento existente sobre o ambiente explorado e pelo posicionamento de cada robô dentro deste ambiente. Por um lado, um robô precisa de informações sobre o ambiente explorado, geralmente por meio de um mapa, para que possa se localizar com relação a essas informações. Por outro lado, um robô precisa saber sua localização no ambiente para que possa construir um mapa.

Nesse sentido, um dos desafios em robótica móvel é prover os dados supracitados enquanto os robôs navegam no ambiente. O problema de localização e mapeamento simultâneos é conhecido como *Simultaneous Localization and Mapping* (SLAM). O problema de SLAM foi originalmente tratado nos trabalhos

de Smith e Cheeseman (1986b) e Smith et al. (1990). Após estes trabalhos, diversas outras abordagens propuseram algoritmos para resolver o problema SLAM. Os principais mecanismos desenvolvidos para resolver o problema de SLAM são baseados nas seguintes abordagens: Filtro de Kalman (*Kalman filter* - KF) (Cho e Kim, 2010; Myung et al., 2010); Filtro de Partículas (*Particle Filters* (Murphy, 2010; Montemerlo et al., 2002a, 2003); e *Graph SLAM* (Yin et al., 2014; Thrun e Montemerlo, 2005; de la Puente e Rodriguez-Losada, 2014)

A estimativa de posicionamento de robô no ambiente explorado é amparada por um conjunto de sensores que se movimentam juntamente com este robô. Tem-se, como exemplo de sensoriamento para estimativa de posicionamento, os sistemas de odometria, que a partir de sinais de controle enviados ao robô podem inferir posicionamento, de forma incremental, tendo como base a localização anterior do robô.

Outros sensores de medida tais como câmeras, sonares e *laser scanners* também podem ser utilizados por um robô, para que informações referentes ao seu posicionamento sejam capturadas e enviadas ao robôs. Os mesmos sensores podem, ainda, capturar e submeter as características do ambiente explorado a um robô. Dessa forma, pode-se construir um mapa a partir da percepção do ambiente e das informações de medida obtidas pelos sensores acoplados ao robô.

Entretanto, Yamauchi et al. (1998) afirma que grande parte das informações obtidas a partir dos sensores será imprecisa, visto que as limitações físicas e outros problemas não podem ser previstos. Assim, as estimativas de mapeamento e localização inferidas pelo robô a partir destas informações apresentarão erros. Infelizmente, os erros mencionados distanciam a realidade e a percepção do ambiente.

Segundo Yamauchi et al. (1998), os erros tendem a se acumular quando não são devidamente tratados, o que causa imprecisão às estimativas. Consequentemente, em um dado momento, o processo de obtenção de estimativas será desnecessário dada a imprecisão que foi acumulada no decorrer do tempo de exploração.

O sensoriamento do ambiente, nesse sentido, deve ser eficaz em problemas nos quais deve-se gerar um mapa. Robôs envolvidos em um problema de busca, por exemplo, dependem de um mapa do local para que possam se localizar no ambiente explorado. Em alguns casos, não existe um mapa do local e os robôs não possuem informações prévias sobre o ambiente que está

sendo explorado. Assim, um mapa deve ser construído ao mesmo tempo em que o robôs tentam se localizar no ambiente.

Soluções probabilísticas, como a de Thrun et al. (2005), tratam o problema de SLAM com a minimização simultânea das incertezas envolvidas nas estimativas. Thrun et al. (2005) assume que ambas estimativas e incertezas são dependentes entre si, e que são necessários resultados de localização precisos para que um mapa preciso seja criado. Nesse sentido, é coerente que a localização e a construção do mapa sejam abordados e resolvidos simultaneamente de forma que possam ser eliminados os erros, impossibilitando o acúmulo destes no decorrer do processo de navegação.

Grande parte dos problemas encontrados em soluções de SLAM estão diretamente relacionados com a precisão e a fidelidade do mapa gerado, uma vez que o posicionamento do robô é inferido utilizando predição de posicionamento. Dado que, muitas vezes, o terreno sobre o qual um robô navega é irregular, erros na predição de posicionamento podem ser gerados.

Ressalta-se que os erros tendem a se acumular a cada iteração da construção do mapa. Por conseguinte, diversos trabalhos restringem a navegação em ambientes fechados e pequenos, tendo como objetivo a diminuição de erros causados por desníveis e deslizes.

Segundo Davison et al. (2007), quando o período de navegação em um ambiente é suficientemente longo, os erros relativos ao posicionamento e à construção de mapas tendem a aumentar. Tal aumento é decorrente da imprecisão dos sensores utilizados ou das condições do local explorado. Consequentemente, os resultados obtidos serão imprecisos e não confiáveis.

Uma vez que os algoritmos SLAM tem como objetivo principal minimizar (ou até mesmo eliminar) erros tratando os problemas de construção do mapa e posicionamento simultaneamente, tem-se como proposta a utilização de informações de uma etapa anterior de navegação na etapa subsequente à esta, visando aumentar a precisão e confiabilidade dos resultados alcançados na etapa anterior. Para tal, Smith e Cheeseman (1986a) propõem o estabelecimento de marcos no ambiente para que o robô possa se localizar neste ambiente em tempo de navegação.

Segundo Burgard et al. (1997), o problema de localização está diretamente ligado com a capacidade que um robô tem de conhecer seu posicionamento e orientação a cada instante em um sistema absoluto de coordenadas. Para Gutmann e Konolige (1999), a criação de mapas tem relação com a capacidade

de atribuição das estruturas que se encontram ao redor do robô, dentro de um mesmo sistema de coordenadas.

Entretanto, nem sempre se consegue fornecer a um robô um mapa confiável do ambiente, visto que tal mapa pode ser alterado no decorrer da navegação ou, até mesmo, nada se sabe sobre o ambiente que será explorado. Ainda, há casos nos quais o ambiente é desconhecido e, consequentemente, não existe um mapa conhecido.

Dessa forma, cabe ao robô as tarefas de exploração do ambiente e construção do mapa. Visto que a posição inicial do robô é desconhecida, fornecer uma localização precisa do robô a cada instante de tempo é uma tarefa complexa. Contudo, até mesmo a utilização de sistemas absolutos de posicionamento pode fornecer resultados não confiáveis, como afirma Negenborn (2003).

Devido a fenômenos não previsíveis e terrenos irregulares, as informações fornecidas ao robô apresentam ruídos, o que prejudica a obtenção de estimativas de localização e criação do mapa. Tais situações são difíceis de serem modeladas, dada a complexidade e imprevisibilidade dos eventos e fenômenos envolvidos. Consequentemente, os erros supracitados tornam imprecisas as estimativas obtidas, de forma que a aplicabilidade destas em algoritmos posteriores é inviável, uma vez que que estes algoritmos dependem de resultados confiáveis e precisos de localização e mapeamento.

Uma alternativa para a erradicação dos referidos erros é a criação de sensores mais precisos que, hipoteticamente, apresentam uma menor taxa de erro, proporcionando resultados mais precisos e confiáveis. Contudo, a aumento de precisão aumenta o preço embutido, fazendo com que tal abordagem deixe de ser atrativa.

No trabalho de Davison et al. (2007) também se argumenta que sensores mais precisos geram grande quantidade de dados, aumentando a complexidade de manipulação dos dados destes sensores. Davison et al. (2007) acrescenta que eliminar os erros desses sensores é impossível, dado que outros fatores também interferem nos resultados, como terreno irregular e demais fenômenos não previsíveis.

Uma vez que erros não podem ser eliminados do problema, têm-se como solução considerá-los na solução do problema de localização e navegação simultâneos. Assim, estes erros são incorporados aos modelos utilizados. Segundo Thrun et al. (2005), tais erros não podem ser previstos individualmente. Mas, quando considerados em conjunto, podem ser tratados estatisticamente, visto

que estes apresentam padrões que podem ser modelados matematicamente. Tal abordagem motivou o surgimento da robótica probabilística, na qual os estados de localização e mapeamento do robô passam a ser distribuições de probabilidades. Estas distribuições indicam os estados possíveis utilizando a precisão das observações realizadas.

Através de uma abordagem probabilística, a minimização das incertezas proporciona a obtenção de estimativas mais precisas. Elfes (1990) propõe a minimização de incerteza a partir do acúmulo de informações, para que os erros aleatórios gerados pelos ruídos dos sensores sejam eliminados.

As observações realizadas por um robô são decorrentes do seu posicionamento no ambiente explorado. Nesse sentido, a sua localização é necessária para a composição de um sistema global de coordenadas, conforme proposto por Thrun et al. (2000). Entretanto, a localização depende diretamente do mapeamento, uma vez que o robô deve ser capaz de identificar as estruturas contidas no ambiente, de forma que se possa obter estimativas absolutas de posicionamento com relação a tais estruturas. Com isso, é possível o robô inferir seu posicionamento no mapa em tempo de navegação.

Se uma das estimativas de posicionamento apresentar erros, estes serão propagados para as próximas iterações da navegação, aumentando a imprecisão e causando o acúmulo de erros. Consequentemente, os resultados finais serão invalidados, tornando necessário lidar com os problemas de posicionamento e mapeamento simultaneamente.

Assim, deve-se ter como foco a minimização das incertezas envolvidas, evitando lidar com estas de forma singular. Dessa forma, os erros gerados pelos sensores em cada uma das iterações podem ser eliminados antes mesmo que se acumulem.

Muitas vezes não há um mapa confiável previamente fornecido para a exploração de um determinado ambiente. Isso se deve ao fato de que tal ambiente pode nunca ter sido explorado ou nem mesmo é conhecido. Nesse cenário, um ou mais robôs devem construir incrementalmente o mapa que será utilizado para a localização no ambiente explorado.

A abordagem apresentada por Smith e Cheeseman (1986a) propõe o estabelecimento de marcos naturais como pontos de referência para localização em um ambiente durante a navegação, sendo estes armazenados em um mapa do ambiente. Este mapa é conhecido pelo robô e, à medida que novos marcos são identificados, estes são incluídos no mapa.

Segundo Betke e Gurvits (1995), quando um marco é reconhecido em um instante posterior, o robô adquire informações relativas à sua própria localização e à posição que ele representa no ambiente. Dessa forma, um robô pode refinar sua própria posição, tornando o processo de mapeamento e localização mais precisos.

Como abordagem para a resolução do problema de localização, Betke e Gurvits (1995) propõe a busca por estruturas fixas no ambiente explorado, que podem ser utilizadas como referências para que o robô possa se localizar. Tais estrutura são conhecidas como *landmarks* (marcos) e devem apresentar propriedades que possibilitem a sua detecção no ambiente.

Para Jensfelt (2001), a capacidade de auto-localização em um ambiente desconhecido é fundamental para qualquer sistema autônomo de navegação. Em tempo de navegação, um robô recebe diversos sinais de controle que possibilitam sua movimentação pelo ambiente. Jensfelt (2001) acrescenta que um robô deve ser capaz de calcular iterativamente quais sinais de controle são necessários para que alcance seu objetivo específico de maneira eficiente e segura.

Nesse sentido, um robô deve ser capaz de monitorar sua localização relativa às estruturas que o cercam, de forma que seja possível obter uma percepção de deslocamento através do ambiente explorado iterativamente.

2.3 Mapeamento de ambientes

Para que um robô consiga executar tarefas quaisquer, muitas vezes faz-se necessária a construção de um mapa, no qual pode-se representar virtualmente um modelo do mundo que o cerca. Por meio deste modelo de mundo, assume-se que um robô pode se localizar, em tempo real, no ambiente explorado.

O conhecimento que um robô tem sobre o ambiente explorado está diretamente relacionado com a percepção dos objetos que estão ao seu redor. Tal conhecimento é essencial para que o robô não colida durante a exploração de um ambiente, que pode ser conhecido ou desconhecido. Tal conhecimento e percepção podem ser adquiridos por meio da utilização de sensores que tem a capacidade de interagir com o ambiente explorado.

Segundo Thrun (2002), um robô pode realizar observações no ambiente explorado em tempo de navegação, podendo decidir, de acordo com os com-

portamentos previamente programados, qual deve ser o seu comportamento mediante as estruturas identificadas no ambiente explorado. É interessante, ainda, que sejam armazenadas as informações relativas à observação do ambiente explorado, para que estas possam ser recuperadas e potencialmente reutilizadas posteriormente.

Estas informações e percepções são comumente armazenadas em um mapa, que é a representação do mundo compreendido pelo robô. As informações e observações coletadas através dos sensores do robô são armazenadas no mapa de forma consistente. Para Pierce e Kuipers (1994), um mapa nada mais é do que um conjunto de variáveis, sendo cada uma destas capazes de descrever estruturas do ambiente de maneira única.

Na construção do mapa, um robô deve determinar a qual região do ambiente as observações realizadas pertencem, para que estas sejam atualizadas no mapa e o estado observado seja então representado. O reconhecimento de regiões pode ser conduzido através do conjunto de características identificadas que cada variável detém, o que a torna única dentro do mapa construído. Davison et al. (2007) ressalta que a quantidade de dados gerados para o mapa depende do tipo de sensor utilizado pelo robô. Sensores mais precisos tendem a gerar uma quantidade maior de informações, o que muitas vezes limita a capacidade de armazenamento de características do mapa construído.

A incerteza nas informações geradas pelos sensores também afetam a construção do mapa. A imprecisão desses sensores podem prejudicar a construção do mapa, fazendo com que uma observação seja atribuída a uma variável errada, ou até mesmo ocasionar a atualização de uma variável com um estado que não corresponde à realidade do ambiente explorado.

Segundo Olson (2002), esse tipo de erro pode ser evitado com a utilização de conjuntos de características mais robustos e erros de atualização do mapa podem ser corrigidos por meio do acúmulo de observações, de forma que os erros aleatórios sejam eliminados e os estados das variáveis envolvidas passem a condizer com o estado real.

Outra definição relevante no processo de construção de mapas é a maneira com a qual um mapa é construído, levando em consideração as diferentes formas de obtenção de informações (sensores). Choset e Nagatani (2001) define que mapas de características e mapas métricos[1] criam uma representação espacial das estruturas que cercam o robô no ambiente, sendo estas distribuí-

[1] Mapa construídos a partir de primitivas geométricas.

das dentro de um sistema global de coordenadas. Já os mapas topológicos[2] abordam a construção de grafos que conectam áreas distintas do ambiente explorado de forma qualitativa.

Os mapas de características são alternativas para o armazenamento das informações capturadas pelos sensores do robô. Segundo Buschka (2006), estes mapas filtram as informações capturadas em busca de padrões relevantes, que são armazenadas na forma de pontos em um sistema de coordenadas.

Os demais dados são descartados do sistema. Nesse sentido, um mapa de características pode ser compreendido como sendo uma estrutura que é composta por um conjunto de características relevantes e uma estimativa de posicionamento que indica coordenadas absolutas destas características no ambiente explorado.

Os sistemas propostos por Davison et al. (2007) e Royer et al. (2007) trabalham com uma abordagem de mapas tridimensionais de características MonoSLAM, proposto por Davison et al. (2007). Para Buschka (2006), a construção de mapas de características propicia a redução do custo computacional, uma vez que quantidades menores de informações é levada em consideração.

Entretanto, a busca por padrões relevantes nas informações manipuladas, bem como a caracterização destes padrões em busca de uma correspondência com estruturas semelhantes, podem acarretar no aumento do custo computacional.

Dado que as variáveis de um mapa de características não possuem posições fixas no espaço observado, a incerteza de estimativa pode ser incorporada diretamente no parâmetro que a define. Nesse caso, utiliza-se uma distribuição probabilística conforme a utilizada no tratamento de problemas de localização. Nesse sentido, entende-se que não há a movimentação das estruturas representadas em cada uma das variáveis e, dessa forma, a distribuição probabilística não se propaga no tempo.

2.4 Visão monocular em problemas de SLAM

Segundo Davison et al. (2007), a utilização de visão computacional em propostas que envolvem o problema de SLAM é escassa. Acredita-se que tal escassez é atribuída à complexidade inerente à resolução de problemas de SLAM

[2]Mapas nos quais a representação de um ambiente é um grafo não dirigido.

utilizando visão computacional. Segundo Royer et al. (2007), a construção de mapas em SLAM tempo real utilizando apenas a visão monocular é uma tarefa difícil.

Grande parte dos trabalhos relacionados utilizam-se de sensores odométricos em conjunto com outros sensores de alta precisão para que a localização dos robôs seja inferida. O trabalho de Wulf et al. (2004) apresenta a combinação de um sensor *laser scanner* 3D em conjunto com um algoritmo SLAM para a produção de um mapa de linhas bidimensional de ambientes internos altamente desordenados.

Newman e Ho (2005) apresentam uma proposta para a resolução de problemas de "*loop-closing*" utilizando, em conjunto, dados obtidos de sensores *laser scanner* e dados visuais dos locais explorados. O problema de "*loop-closing*" consiste na tarefa de decidir se um veículo já retornou a um local já visitado anteriormente.

Vários dos trabalhos que abordam a visão computacional também utilizam-se de câmeras estereoscópicas, por meio das quais é possível se obter dados de profundidade na imagem capturada. O trabalho de Elinas et al. (2006), por exemplo, trata o problema de SLAM em ambiente *indoor* utilizando o filtro de partículas *Rao-Blackwellised* em conjunto com robôs equipados somente com câmeras estereoscópicas. No presente trabalho, destaca-se a utilização de câmeras monoculares convencionais, que não provém dados de profundidade.

A resolução de problemas de SLAM em tempo real utilizando apenas visão monocular foi inicialmente alcançada por Davison (2003). Davison et al. (2007) aprimora a proposta de Davison (2003), apresentando o sistema denominado MonoSLAM, que é um sistema em tempo real capaz de recuperar a trajetória tridimensional de uma câmera monocular que se movimenta constantemente em um espaço desconhecido. O sistema MonoSLAM é a primeira tentativa que obteve sucesso na aplicação SLAM em problemas de robótica móvel utilizando o domínio de visão pura, alcançando desempenho satisfatório.

Entretanto, o trabalho de Davison et al. (2007) apresenta limitações quanto ao ambiente de exploração, uma vez que se comporta bem em pequenos ambiente fechados, nos quais existam no máximo cerca de 100 marcos (características) identificáveis.

O trabalho de Royer et al. (2007) também propõe uma solução SLAM em tempo real. Porém, o sistema proposto por Royer et al. (2007) utiliza-se de um

mapa de características (marcos) construído inicialmente por um robô guiado por um usuário através de uma trajetória qualquer. O robô é capaz de seguir a mesma trajetória de forma autônoma, comparando o mapa anteriormente gerado com as imagens capturadas em tempo real. O sistema proposto utiliza-se apenas de visão monocular. A proposta de Royer et al. (2007) não utiliza de marcos artificiais, assim como a proposta de Davison (2003).

É importante informar que a proposta abordada nesta dissertação considera ambientes internos. Em trabalhos futuros, outros ambientes poderão ser considerados. Diferente do que é proposto por Royer et al. (2007), a presente dissertação propõe a navegação autônoma para solucionar o problema de SLAM utilizando múltiplos robôs, sem o auxílio externo e a interferência humana.

2.5 Robótica colaborativa e estratégias de colaboração

Tem-se como proposta, no presente trabalho, a extensão sistema MonoSLAM desenvolvido por Davison et al. (2007), de forma que ambientes desconhecidos possam ser explorados por uma equipe de robôs. Para que os robôs explorem um ambiente de forma colaborativa, faz-se necessária a existência de uma estratégia de colaboração.

O trabalho de Simmons et al. (2000) aborda o problema de exploração e mapeamento de ambientes desconhecidos de forma colaborativa. O algoritmo de mapeamento utilizado tem como abordagem a maximização de verossimilhança utilizando o algoritmo *Hill Climbing* para encontrar mapas que são condizentes com os dados dos sensores utilizados e odometria. A implementação do algoritmo proposto por Simmons et al. (2000) explicita coordenadas de posicionamento aos robôs, tendo como base as estimativas de ganho de informação esperado e de custo da exploração.

Simmons et al. (2000) tem como objetivo a maximização do ganho de informação esperada, ou seja, o conhecimento sobre o mapa construído iterativamente pelos robôs em tempo real. Nesse sentido, é proposta uma técnica de baixo custo computacional que alcança bons resultados na prática. Contudo, uma solução ótima é computacionalmente inviável.

Na referida técnica, cada robô cria propostas de movimentação com esti-

mativas de custo de locomoção para diversas localizações no ambiente. As propostas são enviadas a uma central de execução que as avalia e atribui tarefas para cada um dos robôs com base nestas. As possíveis sobreposições são levadas em consideração.

Quando um robô opta por visitar uma localização específica, a unidade de execução pode atribuir a um robô uma localização diferente da pretendida pelo robô, visto que a nova posição pode proporcionar maior ganho de informação. Quando os mapas são atualizados, os robôs enviam novas propostas de localização, que podem ainda ser convertidas em novas tarefas. Segundo Simmons et al. (2000), a exploração do ambiente finaliza quando não há mais informações úteis a serem adquiridas.

Burgard et al. (2000) propõe uma técnica probabilística para a coordenação de robôs enquanto estes exploram e constroem um mapa do local explorado. A ideia central da técnica proposta leva em consideração o custo de se alcançar um local distante ainda não alcançado e sua utilidade. Tal utilidade depende da probabilidade de que este local distante seja visível a partir de uma coordenada atribuída a um outro robô.

Quando um robô apresenta melhor custo-benefício entre a utilidade do local e o custo para alcançar este local, o local alvo é atribuído a este robô. Entretanto, tem-se como desafio o estabelecimento apropriado de pontos distintos que servirão como pontos de partida para que os robôs explorem regiões distintas.

Quando um ponto alvo é atribuído a um robô, a utilidade de áreas não exploradas visíveis é reduzida para os demais robôs envolvidos. Dessa forma, pontos alvos distintos são atribuídos a robôs distintos. Nesse sentido, pretende-se adaptar tal abordagem para que o tempo de exploração colaborativa reduza a imprecisão do mapa gerado, aumente a capacidade de exploração do ambiente e diminua o tempo necessário para a exploração de um ambiente desconhecido.

Thrun et al. (2000) apresentam um algoritmo para a construção, em tempo real, de mapas em ambientes internos cíclicos. O algoritmo apresentado combina ideias oriundas de mapeamento incremental (construção de mapas incrementais, máximo verossimilhança) e abordagens não-incrementais (estimativa posterior, correção inversa), podendo também ser estendido para mapeamento multi-agente e mapeamento tridimensional.

O algoritmo de Thrun et al. (2000), por meio da utilização da abordagem

de estimativas posteriores, torna possível a integração de dados coletados por mais de um robô. Dessa forma, permite-se que os robôs envolvidos possam se localizar em um mapa construído por um outro robô. Assume-se que a posição inicial de um robô com relação aos demais é desconhecida, mas que cada robô possui uma posição relativa dentro do mapa de um robô líder de equipe. Assim, para que um único mapa seja gerado, cada robô deve se localizar no mapa do robô líder. Os resultados obtidos apontam que a abordagem utilizada tem um bom desempenho mesmo com a ausência de dados odométricos. Entretanto, os robôs não utilizam câmeras como sensores, mas sim *laser scanners*.

O trabalho de Newman et al. (2002) apresenta uma implementação em tempo real baseada na identificação de características para o mapeamento e localização concorrentes de ambientes internos dinâmicos. Nestas proposta, utiliza-se estatística para a extração de segmentos de linhas dos dados capturados pelo *laser* em tempo real e um método de planejamento para a execução de trajetórias autônomas. A abordagem busca mostrar que um robô pode, de forma autônoma, retornar à sua posição inicial com poucos centímetros de precisão, mesmo com a presença de várias pessoas caminhando no ambiente em questão. Porém,tal abordagem não contempla construção colaborativa do mapa. Os sensores utilizados foram um *laser scanner* e codificadores de rodas e o ambiente no qual os testes foram executados era relativamente homogêneo, com algumas deformações no piso, o que causava, em dados momentos, a derrapagem das rodas do robô utilizado.

Montemerlo et al. (2002b) apresentam o algoritmo FastSLAM, que estima recursivamente a distribuição posterior completa sobre o posicionamento de um robô em escala logarítmica com o número de marcos identificados no mapa. No algoritmo FastSLAM, o problema de SLAM é decomposto em um problema de localização do robô e em uma coleção de estimação de marcos condicionados à estima de posicionamento deste. FastSLAM também utiliza um filtro de partícula modificado para estimar posicionamento posterior nos caminhos dos robôs.

Deve-se ressaltar que, em contraste com a complexidade quadrática identificada em algoritmos SLAM com abordagens baseadas na utilização de *Extended Kalman-filter*, o algoritmo proposto por Montemerlo et al. (2002b) desenvolve uma estrutura de dados que reduz o tempo de execução para ($O(M \log K)$), onde M é o número de partículas no filtro de partículas e K é o número de

marcos identificados. Entretanto, a abordagem utiliza *laser scanners* como sensores e não suporta a construção colaborativa do mapa.

Na proposta de Davison et al. (2007) é apresentado o algoritmo MonoSLAM capaz de recuperar a trajetória tridimensional de uma câmera monocular em tempo real com *frame-rates* altos (tipicamente 30Hz). Para tal, utiliza-se câmeras convencionais (monoculares) como único sensor para a representação e estimativa de posicionamento em ambientes dinâmicos. O ponto positivo desta proposta é a viabilização da implementação em tempo real da estimação probabilística do estado da câmera em movimento e seu mapa, o que permite resolver o problema de SLAM eficientemente.

A proposta de Davison et al. (2007) tem como um dos objetivos específicos a simplificação do *hardware* necessário para a implementação de SLAM baseado em visão computacional, ou seja, reduzir a plataforma para a utilização de uma única câmera conectada a um computador, requerendo ainda o mínimo possível de suposições sobre a movimentação tridimensional da câmera.

O algoritmo MonoSLAM alcança a eficiência desejada em operações em tempo real utilizando uma abordagem guiada pelo mapeamento de medições de características (*features*). Tal abordagem consiste em um modelo de movimentação generalizado para movimentos tridimensionais em câmeras 3D para capturar as informações dinâmicas prioritárias em *streams* contínuos de vídeo.

Embora a localização e o mapeamento estejam diretamente ligados, os autores direcionam esforços nos problemas relacionados à localização. Um mapa é corretamente construído, porém, o mapa construído é um mapa esparso de marcos otimizados que permitem a localização no ambiente. Todavia, os mapas não são construídos de forma colaborativa.

Outro aspecto importante é que muitos dos trabalhos relacionados citados por Davison et al. (2007) utiliza técnicas de SLAM baseadas em visão estéreo. Um deles é o trabalho de Neira et al. (1997), cujo sistema SLAM foi o primeiro a utilizar a visão estéreo com processamento em tempo real (5Hz), capaz de construir um mapa tridimensional de marcos naturais controlando um robô.

Zou (2013) introduzem uma proposta de estimação de posicionamento e mapeamento para localização e mapeamento (SLAM) de objetos dinâmicos utilizando múltiplas câmeras. Tais câmeras se movem independentemente, podendo também ser montadas em diferentes plataformas. As câmeras utilizadas atuam juntas na construção de um mapa global, que inclui o posicio-

namento tridimensional de pontos estáticos e trajetórias de pontos em movimento.

Nesta proposta manteve-se a incerteza de posicionamento de cada ponto no mapa para que fosse aumentada a robustez do sistema. As câmeras utilizadas são agrupadas de acordo com a sobreposição de visão. Também são gerenciadas as divisões e as fusões dos grupos de câmeras em tempo real. Os experimentos mostraram que o sistema pode funcionar de forma robusta em ambientes dinâmicos e produzir resultados precisos em ambientes estáticos. Entretanto, o sistema implementado trabalha *offline* com dados de vídeos pré-capturados.

Um algoritmo para a localização de múltiplos robôs é descrito por Han et al. (2011). Os autores utilizam o algoritmo KF (*Kalman filtering*) para localizar cada robô de acordo com suas velocidades e informações de posicionamento. Os robôs são equipados com câmeras para identificar marcos naturais e corrigir seus posicionamentos.

Para a resolução do problema SLAM considerando múltiplos robôs, o mapa deve ser gerado por meio da fusão dos mapas locais obtidos por cada robô. No trabalho de Ballesta et al. (2010), a fusão do mapa é dividida em um problema de alinhamento e um problema de junção. No problema de alinhamento, calcula-se a transformação entre os mapas individuais com diferentes sistemas de coordenadas. Em seguida, os dados alinhados devem ser mesclados para gerar um mapa global.

No trabalho apresentado por Forster et al. (2013), Veículos Aéreos Não-Tripulados (VANT) detectam *features* de um ambiente gerando mapas individuais. Os mapas são mesclados sempre que sobreposições são detectadas. Deste modo, os veículos pode estimar seu posicionamento em um mapa global.

Embora existam várias abordagens para resolver o problema SLAM utilizando múltiplos robôs, a maioria deles não apresenta um mecanismo de navegação autônoma. Nestes casos, o problema SLAM geralmente é resolvido utilizando câmeras portáteis. Em alguns casos, a navegação é baseada apenas em formulação matemática.

Assim, os robôs são muito dependentes de parâmetros e o desempenho do sistema pode sofrer degradação crítica devido às falhas dos robôs. Teorias bio-inspiradas e evolucionistas fornecem fundamentos para projetar estratégias alternativas. Particularmente, as versões artificiais análogas a mecanismos

biológicos que definem uma dinâmica de organização social, observados em alguns sistemas de enxames biológicos, são muito apropriados em aplicações que envolvam múltiplos agentes.

2.6 Considerações finais

Neste capítulo, apresentou-se uma revisão de trabalhos selecionados nas seguintes áreas de estudo: Localização e mapeamento simultâneos, mapeamento ambientes, visão monocular em problemas de SLAM e robótica colaborativa.

Considerando os trabalhos abordados anteriormente, a presente dissertação propõe a extensão do problema SLAM para múltiplos robôs, na qual os robôs são capazes de navegar de forma autônoma. A movimentação dos robôs tem como base os dados de odometria visual obtidos a partir da extensão do algoritmo MonoSLAM. A navegação autônoma é provida a partir de técnicas bio-inspiradas que não possuem um conhecimento prévio do ambiente.

CAPÍTULO

3

Localização e mapeamento simultâneos utilizando visão monocular

3.1 Considerações iniciais

Diversas abordagens presentes na literatura resolvem o problema de localização e mapeamento simultâneos através da utilização de sensores *laser scanners* e sensores odométricos, sendo estes utilizados em conjunto ou de forma isolada. Os trabalhos de Simmons et al. (2000), Montemerlo et al. (2002b), Newman et al. (2002), Wulf et al. (2004) e Newman e Ho (2005) são exemplos que utilizam sensores *laser scanners* como sensores para estimativa de posicionamento.

Contudo, na abordagem proposta nesta dissertação, adotou-se que uma equipe de robôs deve atuar de forma autônoma, sem a intervenção ou o auxílio externo, utilizando apenas o campo de visão pura como sensor para estimativa de posicionamento. Assim, para que os robôs consigam se locomover e, ao mesmo tempo, se localizar no ambiente explorado, os problemas de localização e mapeamento do ambiente devem ser resolvidos simultaneamente.

Consequentemente, um dos grandes problemas que devem ser resolvidos

em estratégias de navegação e exploração colaborativa é o problema de auto-localização dos robôs envolvidos. O problema de auto-localização consiste em estimar uma posição relativa dentro do ambiente explorado. Cada um dos robôs que atuam em uma tarefa deve conhecer seu próprio posicionamento no ambiente e obter informações sobre o posicionamento dos demais robôs envolvidos na mesma tarefa.

Como as tarefas devem ser cumpridas de forma colaborativa, é importante que cada um dos robôs saiba se localizar com relação aos demais robôs para que as tarefas possam ser cumpridas em conjunto. Contudo, no início da exploração de um ambiente parcialmente ou completamente desconhecido, um robô não possui informações referentes à sua localização, nem mesmo conhece um mapa do local. Assim, iterativamente, cada um dos robôs deve estimar seu posicionamento no ambiente explorado de acordo com a movimentação realizada.

Analisando os avanços no campo de visão pura, observou-se que é crescente o número de trabalhos que abordam aplicações VSLAM (SLAM visual) utilizando sensores primariamente visuais (Mozos et al., 2007; Lee e Song, 2008; Kootstra e Schomaker, 2009; Kundu et al., 2010; Clipp et al., 2010; Williams e Reid, 2010; Lategahn et al., 2011; Lee e Lee, 2013; Johannsson et al., 2013). Karlsson et al. (2005) identificaram que aplicações VSLAM são estudadas intensamente, uma vez que é crescente o número de dispositivos diversos equipados com câmeras de vídeo. Considerando situações nas quais todos os robôs que compõem um time possuem sua própria câmera de vídeo acoplada, assumiu-se, no presente trabalho, que cada robô deve ser capaz de estimar (predizer) sua própria localização relativa dentro do ambiente explorado utilizando apenas o campo de visão pura.

Nesse sentido, visando resolver o problema de auto-localização utilizando apenas o campo de visão pura, optou-se pela implementação do algoritmo MonoSLAM, proposto por Davison et al. (2007). Nesta dissertação, o MonoSLAM é adotado estritamente para prover dados de odometria visual para sistemas multi-robôs (SMR).

Este capítulo é organizado como segue. Na Seção 3.2, apresenta-se uma série de conceitos teóricos e a formulação do problema. Na Seção 3.3, detalhes de implementação são apresentados. Os experimentos e resultados obtidos durante a execução dos experimentos realizados são discutidos na Seção 3.4. Por fim, na Seção 3.5, as considerações finais deste capítulo são fornecidas.

3.2 Algoritmo MonoSLAM

MonoSLAM é um sistema SLAM baseado no campo de visão pura, capaz de estimar, em tempo real, a trajetória tridimensional de uma câmera monocular em um ambiente desconhecido. No presente trabalho, tem-se como proposta que cada um dos robôs integrantes de uma equipe de robôs execute sua própria instância do MonoSLAM e seja capaz de construir um mapa probabilístico tridimensional de *features*.

Nesse sentido, estendeu-se o sistema MonoSLAM desenvolvido por Davison et al. (2007), de forma que os robôs empregados na resolução de tarefas de exploração de ambientes desconhecidos sejam capazes de construir, colaborativamente, um mapa tridimensional de características utilizando apenas o domínio de visão pura como sensor para estimativa de posicionamento.

O mapa supracitado deve ser representado de acordo com os mapas tridimensionais de características gerados no sistema MonoSLAM. Para isso, Davison et al. (2007) propõem um mapa probabilístico tridimensional baseado em características salientes identificadas no ambiente monitorado. Nesse mapa são representados, a cada instante, a posição estimada da câmera utilizada pelo agente, as posições estimadas das características de interesse e a incerteza dessas estimativas. O mapa probabilístico tridimensional é um conceito chave na abordagem MonoSLAM, uma vez que representa, a todo instante, as estimativas atuais do estado da câmara e todas as *features* de interesse capturadas no ambiente explorado.

A partir da inicialização do sistema, o mapa probabilístico é continuamente atualizado através da aplicação do Filtro de Kalman Estendido (*Extended Kalman Filter - EKF*). Durante a movimentação da câmera, as *features* salientes são observadas e um estado probabilístico de estimativa da câmera é continuamente atualizado. Novas *features* identificadas são armazenadas no mapa probabilístico tridimensional com os novos estados atualizados. Quando necessário, algumas *features* podem ser removidas do mapa em questão.

O mapa probabilístico mantido no MonoSLAM é matematicamente representado por um vetor $\hat{x}$ e uma matriz de covariância M, onde $\hat{x}$ é composto pelas estimativas de estado das *features* e da câmera e M é uma matriz qua-

drada que pode ser particionada em novos sub-elementos de matriz:

$$\hat{x} = \begin{pmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \\ \vdots \end{pmatrix}, M = \begin{bmatrix} P_{xx} & P_{xy1} & P_{xy2} & \dots \\ P_{y1x} & P_{y1y1} & P_{y1y2} & \dots \\ P_{y2x} & P_{y2y1} & P_{y2y2} & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3.1}$$

O vetor que representa o estado da câmera $\hat{x}_v$ é representado por um vetor r^W, um quaternion[1] q^{RW}, um vetor de velocidade v^w e um vetor de velocidade angular ω^R, onde W é uma estrutura que representa o ambiente explorado e R é um quadro de imagem obtido da câmera, conforme observado na Equação 3.2.

$$\hat{x}_v = \begin{pmatrix} r^W \\ q^{WR} \\ v^W \\ \omega^W \end{pmatrix} \tag{3.2}$$

O mapa mantido pelo MonoSLAM não tem como objetivo descrever o ambiente explorado por completo, mas sim prover um referencial para que a localização da câmera possa ser estimada em tempo real. Com isso, o algoritmo visa capturar um conjunto esparso de pontos referenciais de alta qualidade (*features* de alta qualidade).

Na Figura 3.1, ilustra-se um quadro do vídeo capturado por uma câmera monocular, utilizando o sistema MonoSLAM, no qual pode-se observar a extração de características salientes (retângulos com bordas vermelhas) e uma representação virtual do mapa probabilístico tridimensional, no qual cada uma das características salientes é representada em um sistema de coordenadas tridimensional. Por meio dessa representação virtual, pode-se medir a distância relativa entre a câmera e as característica identificadas.

Davison et al. (2007) considera a cena do ambiente como um universo rígido, no qual cada marco é uma *feature* estacionária no ambiente. Nesse sentido, assume-se que cada marco é uma *feature* com um ponto de localização no espaço tridimensional. A câmera é modelada como um corpo rígido que pode ser rotacionado e transladado, que apresenta uma localização no espaço tridimensional explorado e que mantém estimativas de velocidade angular e

[1] O termo quaternion se refere à uma notação matemática adequada para a representação de orientações e rotações de objetos em 3 dimensões.

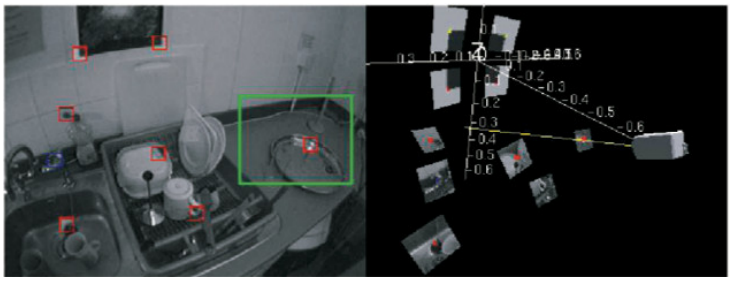


Figura 3.1: Visualização de características salientes e mapa probabilístico 3D.Extraída de: Davison et al. (2007)

linear.

O tamanho total de representação do mapa está em ordem de $O(N^2)$, onde N é o número de *features* armazenadas. A execução do algoritmo MonoSLAM também está na ordem de complexidade $O(N^2)$. Com isso, tem-se um conjunto f_i com um número limitado de *features*. Davison et al. (2007) estima que o número de *features* mantido é limitado a cerca de 100, considerando uma implementação que processa *frames* de imagem a 30Hz.

Na abordagem do MonoSLAM, as *features* são descritas como variáveis Gaussianas. O estado do sistema, identificado pela estimativa de posicionamento da câmera e pelo mapa construído, é representado no instante $t = k\Delta t$, onde Δt é o tempo decorrido desde a iteração anterior da câmera k, como variável estocástica com distribuição Gaussiana

$$\hat{\mu}_k \sim N(\mu_k, \Sigma_k) \tag{3.3}$$

com razão μ_k e covariância Σ_k.

A função matemática que representa a estimativa de evolução do estado da câmera é dada por

$$x_{k+1|k} = \begin{pmatrix} \hat{r}^W_{novo} \\ \hat{q}^{WR}_{novo} \\ \hat{v}^W_{novo} \\ \hat{\omega}^W_{novo} \end{pmatrix} = \begin{pmatrix} r^W + (v^W + V^W)\Delta t \\ q^{WR} \times q((\omega^R + \Omega^R)\Delta t) \\ v^W + V^W \\ \omega^R + \Omega^R \end{pmatrix} \tag{3.4}$$

onde a notação $q((\omega^R + \Omega^R)\Delta t)$ representa um quaternion definido pelo vetor de rotação $(\omega^R + \Omega^R)\Delta t$.

O passo de atualização EKF, necessário para as medições no ambiente ex-

plorado, é matematicamente representado por

$$h(\mu) = \begin{pmatrix} h(f_1, x_k) \\ \vdots \\ h(f_m, x_k) \end{pmatrix} \tag{3.5}$$

onde a função de projeção h é capaz de projetar as *features* capturadas em um plano tridimensional, utilizando a estimativa de posicionamento da câmera.

3.3 Implementação do algoritmo MonoSLAM

Visando prover informações de odometria visual para sistemas multi-robôs, uma implementação existente do algoritmo MonoSLAM, proposta por Russo et al. (2014), foi modificada e adaptada para a abordagem proposta. A implementação teve como base o ROS (Robot Operating System) [2], a linguagem `C++` e o sistema operacional `Ubuntu Linux 14.04`.

O ROS é um *framework* flexível utilizado para a implementação de *softwares* para robôs. Trata-se de uma coleção de bibliotecas, ferramentas e convenções que visam simplificar a criação de comportamentos robustos e complexos para robôs de diversas plataformas. Considerando tais características, adotou-se o ROS como *framework* para implementação do algoritmo MonoSLAM.

3.3.1 Arquitetura de implementação do MonoSLAM

A arquitetura de implementação proposta para o MonoSLAM sobre o ROS é descrita na Figura 3.2. Cada novo quadro (*frame*) de imagem f_k é obtido da câmera através do driver `USB Video Class (UVC)` no ROS. Os *frames* capturados são pré-processados e então submetidos ao nó de execução do MonoSLAM, para que o posicionamento e a movimentação da câmera sejam estimados e o mapa de *features* seja construído e atualizado continuamente.

[2] http://www.ros.org

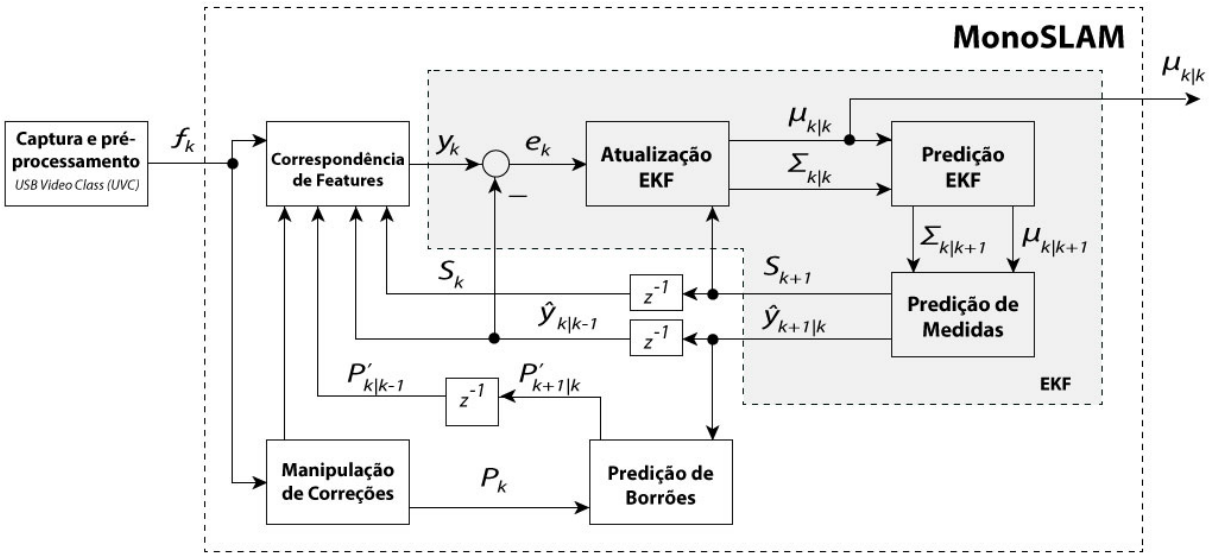


Figura 3.2: Arquitetura definida para a implementação do MonoSLAM no ROS.

A saída de execução μ_k do nó MonoSLAM a cada passo de iteração é o estado estimado do sistema,

$$\mu_k = \begin{pmatrix} x_k \\ f_1 \\ \vdots \\ f_m \end{pmatrix} \tag{3.6}$$

onde μ_k é um vetor que encapsula as informações sobre as *features* capturadas e estimativa de posicionamento da câmera k, considerando um conjunto de *features* $f = \{1, 2, \ldots, m\}$.

Após o pré-processamento e submissão do *frame* f_k, o bloco de correspondência de *features* tem como função buscar a correspondência existente entre um novo *frame* e o conjunto de *features* contido em $P'_{k|k-1}$ (saída do bloco de predição de borrões na Figura 3.2).

A implementação de Russo et al. (2014) utiliza uma técnica para melhorar o desempenho no processo de correspondência supracitado. Tal técnica se resume a uma busca por novas *features* correspondentes somente em um espaço cuja probabilidade de se capturar novas ocorrências é maior.

O bloco de correspondências utiliza informações de medidas previamente estimadas e sua covariância relacionada $S_{k|k-1}$ para o cálculo do vetor y_k. Quando o vetor y_k é calculado, os passos de atualização e estimativa com EKF

são executados. Com isso, um vetor de inovação e_k atualizado é computado. A Equação 3.7 demonstra matematicamente o vetor de atualização obtido:

$$e_k = y_k - y_{k|k-1} \tag{3.7}$$

Os passos de atualização e predição (estimativa) EKF são executados para que sejam estimados, respetivamente, o estado atualizado do sistema $\mu_{k|k}$ com sua covariância $\Sigma_{k+1|k}$ (saída do bloco *Atualização EKF* na Figura 3.2).

Já o bloco de manipulação de correções tem como objetivo gerenciar o conjunto de *features* P_k, para que sejam encontradas as melhores *features* a serem seguidas durante a inicialização do sistema e as *features* antigas que não são mais úteis e podem, eventualmente, ser removidas do mapa.

As correções de estimativa sobre borrões nas imagens, que são gerados por decorrência da rápida movimentação da câmera, são realizadas no bloco de correções de borrões. Um outro bloco importante na arquitetura de implementação é o bloco Z^{-1}. Este bloco tem como objetivo representar um tempo de atraso igual ao inverso da taxa de *frames* da câmara. O bloco Z^{-1} é utilizado para armazenar informações de estimativa até que um novo *frame* f_k seja submetido.

No ROS, um nó (*node*) é um processo que realiza operações específicas de cálculo. Os nós podem ser combinados em um grafo e são capazes de se comunicarem entre si através de tópicos (*topic*) de comunicação. Um sistema para controle de robôs geralmente concentra vários nós e tópicos. Exemplificando, pode-se definir uma arquitetura na qual os seguintes nós se comunicam: um nó para o controle de sensores infravermelho; um nó para o controle dos motores das rodas do robô; um nó para prover mecanismos de planejamento de rotas; e um nó para prover a visualização gráfica do sistema como um todo.

A Figura 3.3 ilustra um grafo do nó de execução do MonoSLAM, bem como os tópicos gerados. O nó MonoSLAM implementado gera tópicos que publicam as seguintes informações: estimativa de posicionamento da câmera em um mapa probabilístico 3D (`\camera_pose`); estimativa de caminho percorrido pela câmera (`\path`); mapa probabilístico 3D contendo as *features* salientes capturadas (`\features`); mapa probabilístico 3D contendo apenas as *features* salientes próximas à câmera (`\close_features`); e um quaternion com dados de odometria (`\odometry`).

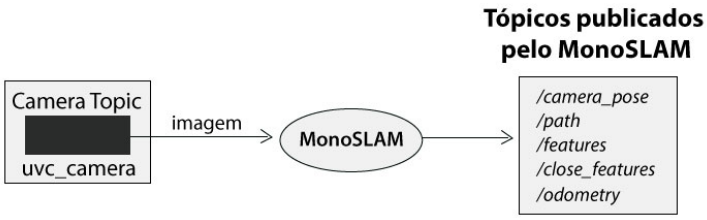


Figura 3.3: Grafo com os tópicos publicados pelo nó MonoSLAM no ROS.

3.3.2 Detalhes de implementação e execução

A implementação do algoritmo MonoSLAM idealizada no presente trabalho considera múltiplas instâncias de execução, uma vez que tem-se como proposta a preparação de um ambiente para múltiplos robôs. Nesse sentido, buscou-se preparar um ambiente de execução adequado, de fácil utilização e flexível, tendo como intuito suportar a execução de tarefas de exploração e vigilância de ambientes controlados.

A execução de múltiplas instâncias do algoritmo proposto ocorre por meio de arquivos de configuração e diretivas de execução em linha de comando. Nesse processo, as instâncias são inicializadas e uma identificação é fornecida para cada um dos robôs que compõem uma mesma equipe. A diretiva de execução do nó MonoSLAM pode ser observada a seguir:

```
$ rosrun monoslam monoslam conf.xml r1 -x 1 -y 4 -d top
```

Na linha de comando descrita anteriormente, executa-se o nó `monoslam` passando o arquivo de configuração da câmera utilizada no formato `XML`, a identificação do robô (`r1`), a posição inicial do robô no mapa (`-x 1 -y 4`) e a direção inicial de movimentação do robô (`-d top`). O arquivo de configurações da câmera (`conf.xml`) contém os parâmetros de calibração que serão utilizadas pelo sistema de visão artificial.

A direção de movimentação inicial é essencial para que as estimativas de posicionamento dos robôs sejam condizentes com a realidade. Isso facilita a montagem de um mapa compartilhado de *features*. Definiu-se, para fins de experimentação, quatro possíveis arranjos iniciais, conforme observado na Figura 3.4. Pretende-se que, em trabalhos futuros, a posição inicial do robôs seja indicada em um arquivo de configuração próprio, no qual será possível inserir diferentes coordenadas e arranjos.

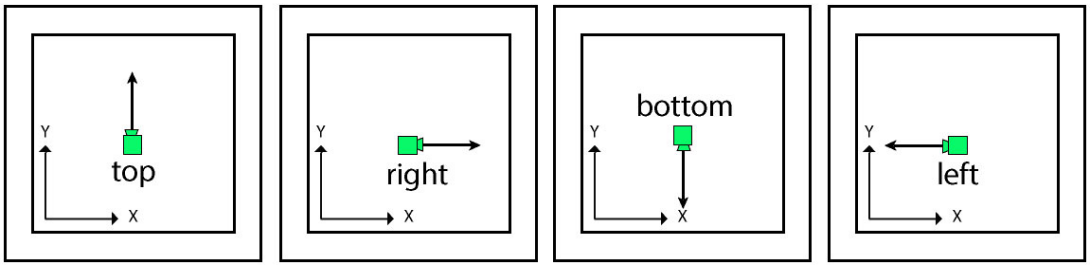


Figura 3.4: Possíveis arranjos de movimentação inicial na execução do nó MonoSLAM no ROS.

Além dos parâmetros de configuração da câmera, as estimativas de posicionamento das *features* capturadas e da câmera são corrigidas de acordo com as posições iniciais fornecidas. Os tópicos gerados por cada instância de execução são nomeados de acordo com a identificação fornecida através da linha de comando.

Dessa forma, na inicialização do sistema, o mapa do ambiente explorado passa a ser construído de forma condizente com o posicionamento real dos robôs. A Figura 3.5 ilustra a arquitetura multi-robôs definida para a execução de múltiplas instâncias do nó MonoSLAM no ROS. Vale ressaltar que, para fins experimentais, apenas uma máquina central é utilizada para executar as instâncias do MonoSLAM.

Considerando a arquitetura definida anteriormente, observa-se que a imagem capturada é transmitida utilizando o driver `USB Video Class (UVC)` no ROS. Dessa forma, cada *frame* de imagem capturado é enviado ao nó de execução do MonoSLAM no ROS. A visualização da execução do algoritmo MonoSLAM pode ser realizada por meio da utilização da ferramenta ROS RViz (Figura 3.6). O RViz é uma ferramenta integrada ao ROS que possibilita a visualização tridimensional dos dados que são publicados pelos nós e tópicos em execução.

Vale ressaltar que o ambiente visual (RViz) foi devidamente preparado e configurado para a visualização dos dados gerados por múltiplas instâncias de execução do nó MonoSLAM, uma vez que o presente trabalho aborda sistemas multi-robôs. Tal configuração pode ser melhor visualizada na Figura 3.7. Nesta, observa-se a execução simultânea de 3 instâncias do nó MonoSLAM. No arquivo de configuração visual gerado no RViz, os tópicos gerados por cada

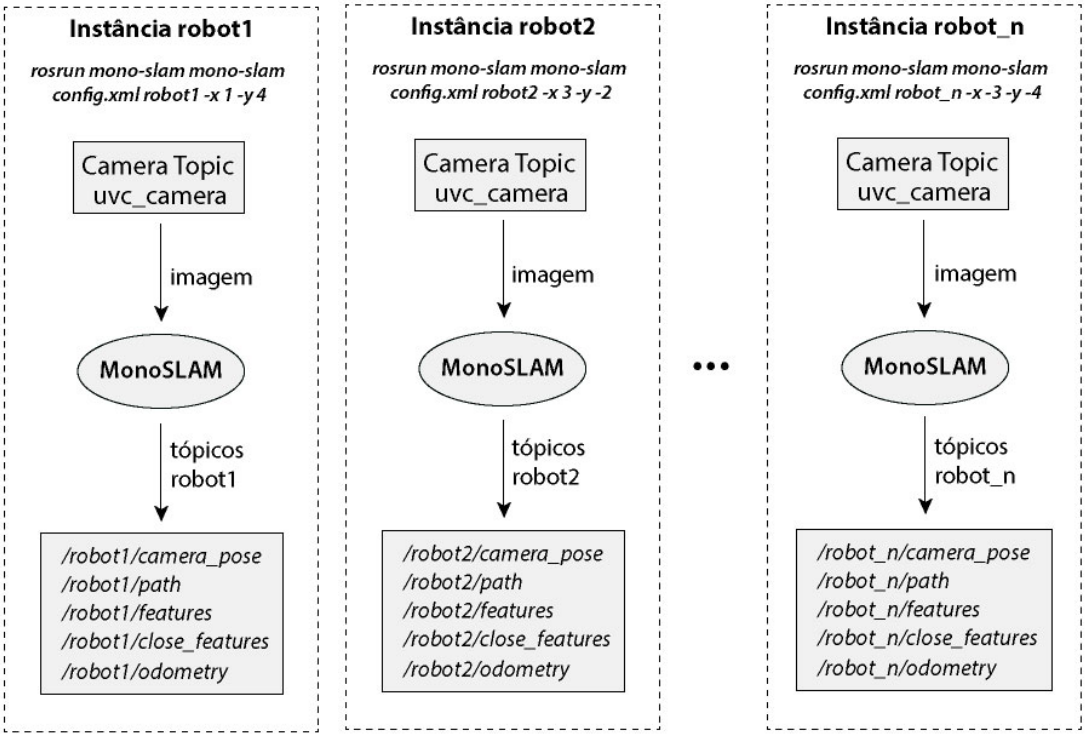


Figura 3.5: Arquitetura multi-robôs para a execução de múltiplas instâncias do nó MonoSLAM no ROS.

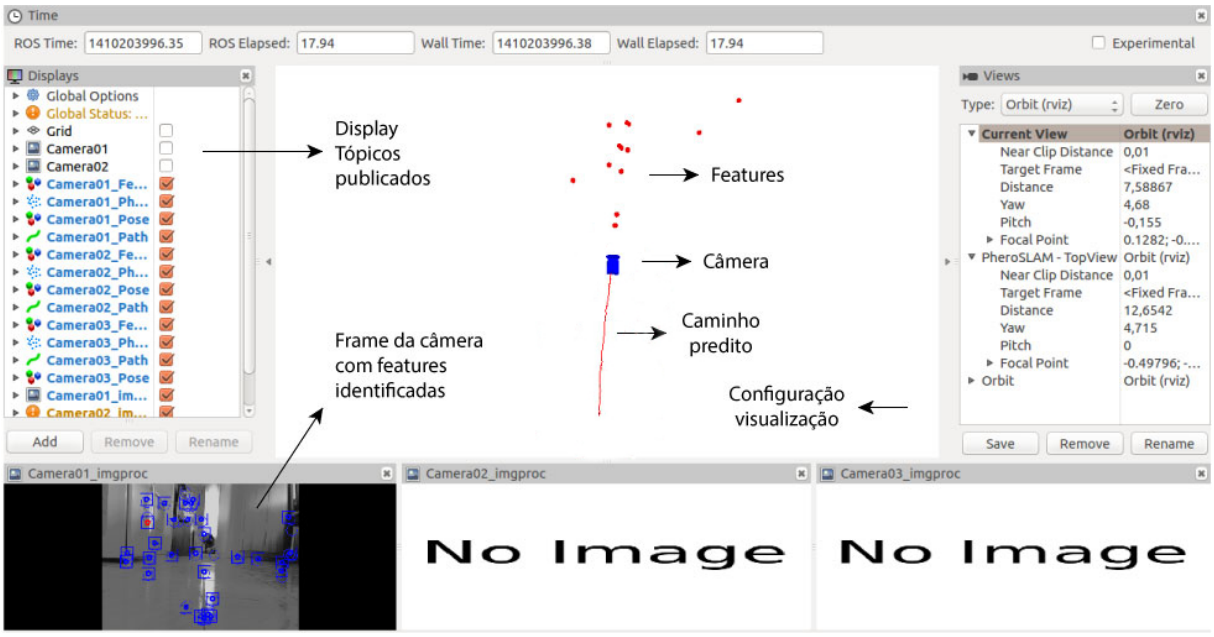


Figura 3.6: Ambiente RViz durante a execução do nó MonoSLAM.

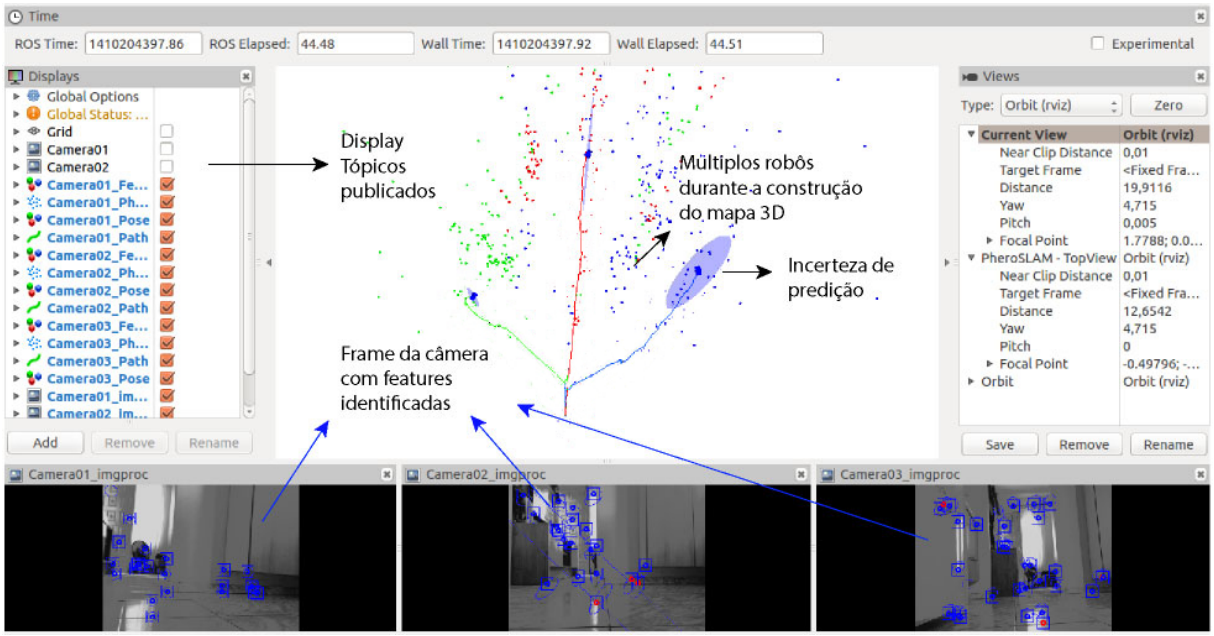


Figura 3.7: Ambiente RViz preparado para a execução múltiplas instâncias do nó MonoSLAM.

instância de execução foram separados e organizados para melhor visualização do estado do sistema. Além das *features* e caminho percorrido pelos robôs, podem ser visualizadas, em tempo real, as incertezas de estimativa (elipsoide em azul na Figura 3.7).

3.4 Resultados experimentais

A fim de validar e testar a robustez da implementação do algoritmo MonoSLAM, alguns experimentos em um ambiente real e em um ambiente simulado foram idealizados. A estratégia de experimentação em um ambiente real é considerada para viabilizar situações reais de exploração em ambientes internos, tendo como foco verificar o comportamento de execução da técnica proposta em escala real de execução. Já a experimentação em ambientes simulados tem como foco a validação do algoritmo MonoSLAM em robôs simulados.

A Figura 3.8 mostra uma visão superior do ambiente utilizado para a experimentação da implementação do algoritmo MonoSLAM. Trata-se de um ambiente real interno com poucos obstáculos. Considerando o referido ambiente, alguns trajetos foram pré-definidos para a execução de tarefas de exploração utilizando o sistema MonoSLAM implementado.

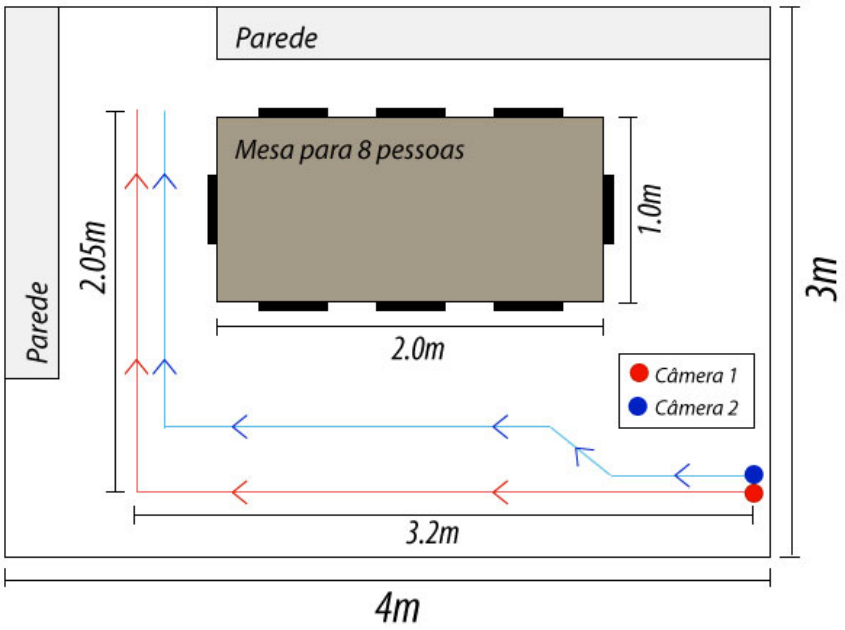


Figura 3.8: Visão superior do ambiente interno explorado utilizando apenas MonoSLAM.

É importante ressaltar que o algoritmo MonoSLAM não tem como propósito guiar robôs em tarefas de exploração de forma autônoma. Trata-se apenas de uma abordagem capaz de capturar *features* relevantes em uma imagem, projetar as *features* capturadas em um mapa probabilístico tridimensional e estimar o posicionamento da câmera em relação às *features* capturadas.

Os trajetos de exploração definidos, bem como a direção de movimentação de cada uma das câmeras estão representados na Figura 3.8. Para a realização do trajetos, definiu-se, como estratégia, acoplar uma câmera monocular de alta resolução à um veículo móvel em miniatura, para então movimentar tal veículo pela trajetória pré-definida. Nos experimentos, utilizou-se a câmera `Logitech HD Pro Webcam C920`.

A Figura 3.9 também mostra a execução do algoritmo MonoSLAM durante a exploração do ambiente descrito anteriormente. O lado esquerdo da Figura 3.9 mostra um *frame* de imagem capturado e processado pelo algoritmo MonoSLAM. Neste quadro, pode-se observar uma demarcação (em azul) relativa às *features* de interesse que foram capturadas pelo algoritmo. À direita, os pontos vermelhos demarcados correspondem à estimativa de posicionamento de cada uma destas *features*. A cada nova iteração do algoritmo MonoSLAM,

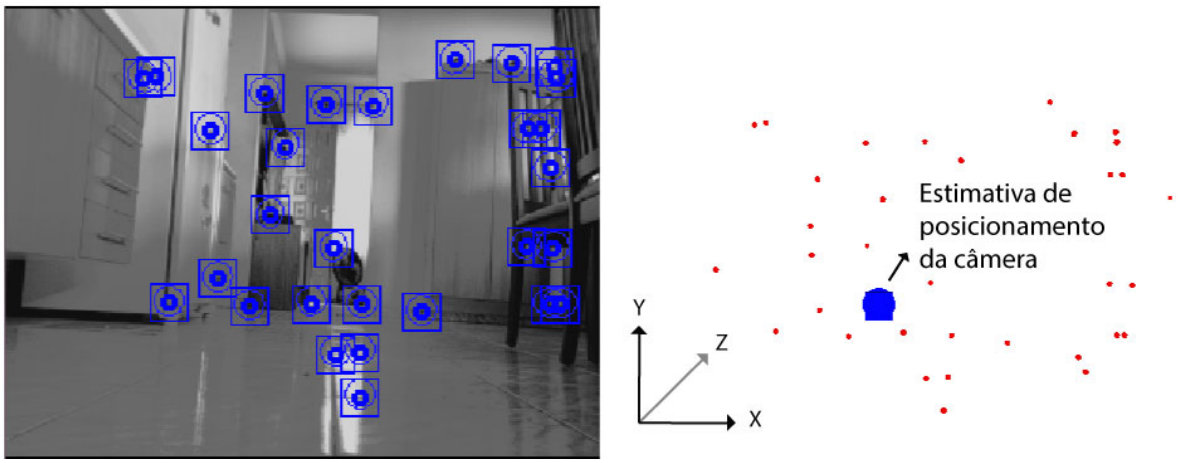


Figura 3.9: Execução do algoritmo MonoSLAM em tempo-real.

novas *features* de interesse podem ser capturadas. Com isso, o mapa probabilístico de *features* é constantemente atualizado com as novas *features* identificadas.

As imagens submetidas para o nó de execução do MonoSLAM são processadas e, nesse processo, as *features* relevantes na imagem são identificadas e armazenadas no mapa probabilístico mantido pelo algoritmo. Simultaneamente, o algoritmo MonoSLAM estima (prediz) o posicionamento da câmera em relação às *features* até então capturadas. A Figura 3.10 mostra uma visão superior do mapa de *features* obtido ao final da execução do MonoSLAM, considerando o trajeto proposto na Figura 3.8.

A Figura 3.11 estabelece um comparativo visual no qual pode-se observar os trajetos propostos para exploração sobrepostos aos trajetos estimados pelo algoritmo MonoSLAM para cada uma das câmeras. Pode-se notar que as trajetórias estimadas apresentam erros de estimativa. Além do caráter probabilístico inerente ao processo de predição (estimativa) e projeção tridimensional, tem-se, como fator relevante para a geração dos erros observados, os desníveis no terreno do ambiente explorado. Tais fatores são responsáveis diretos pela geração de erros acumulados que alteram ligeiramente a trajetória esperada.

Os dados de execução, como a quantidade de *features* capturadas e a distância percorrida por cada agente foram tabulados e podem ser observados

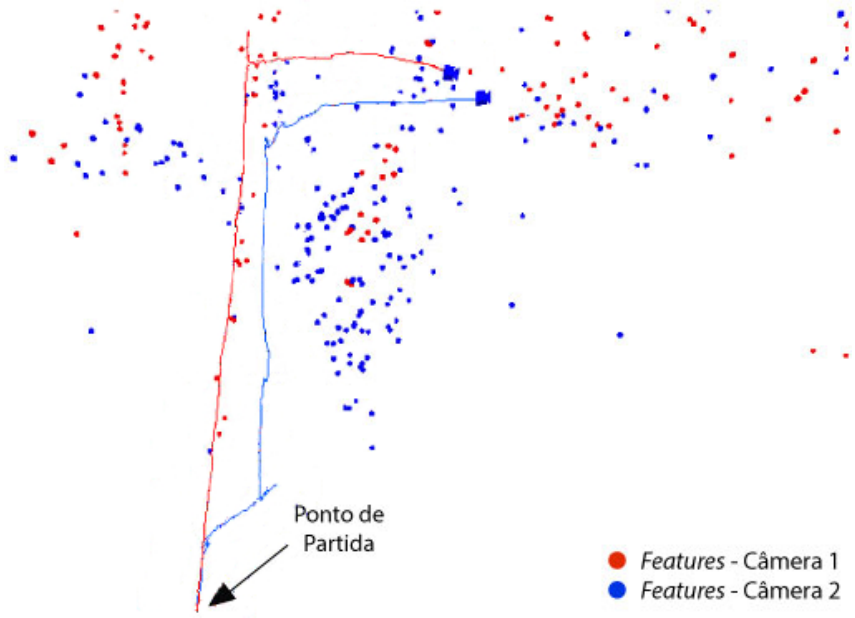


Figura 3.10: Imagem obtida no ROS RViz mostrando a visão superior dos trajetos realizados por cada um dos robôs, executando apenas MonoSLAM. Os pontos azuis e vermelhos representam as *features* capturadas por cada um dos robôs.

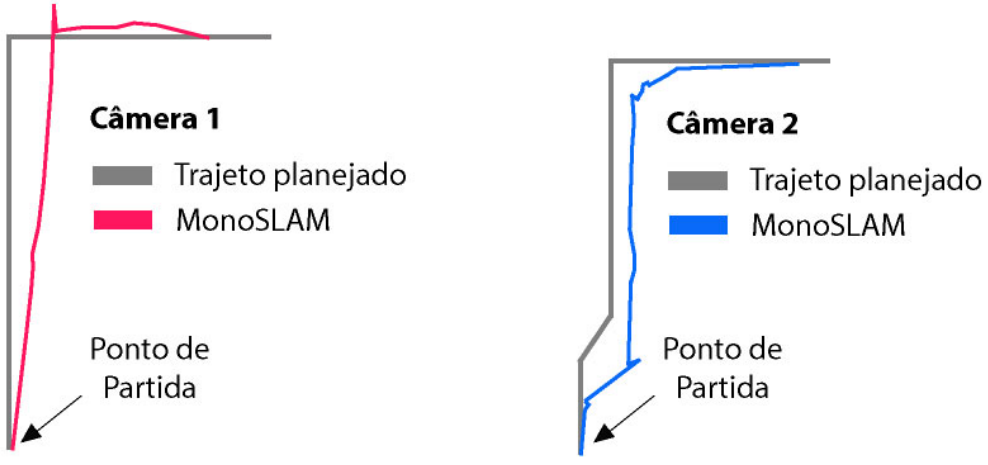


Figura 3.11: Projeção da visão superior dos trajetos planejados e efetivamente realizados por cada um dos robôs utilizando MonoSLAM.

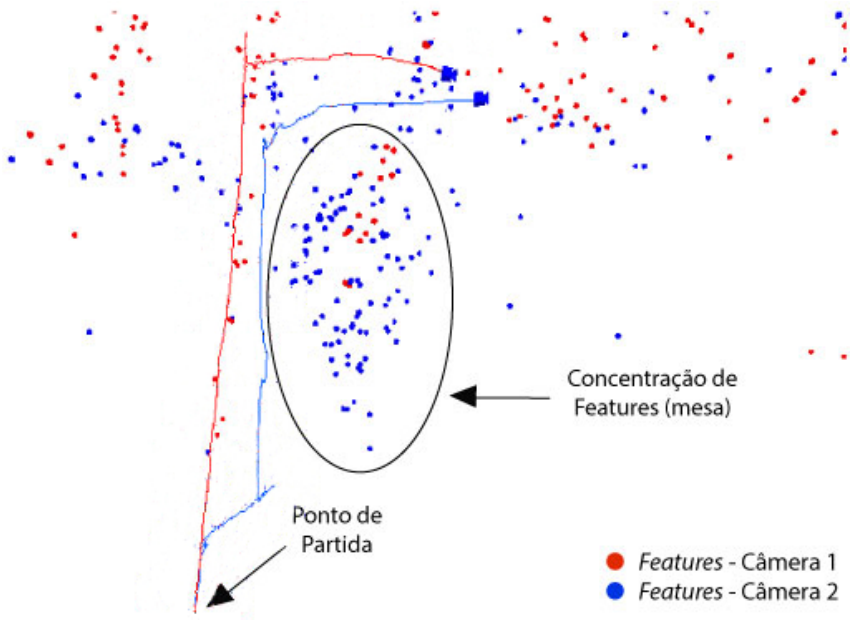


Figura 3.12: Concentração de características (*features*) que denota a presença de um grande obstáculo.

na Tabela 3.1. Os dados em questão são condizentes com os experimentos demonstrados na Figura 3.10. Na figura em questão, algumas das *features* capturadas foram omitidas por motivo de visualização.

Câmeras	**Quant. de *features* capturadas**	**Quant. de *features* removidas**	**Dist. percorrida/ Dist. trajeto (metros)**
Câmera 1	174	80	4.96/5.25
Câmera 2	198	101	4.74/4.92

Tabela 3.1: Relação entre *features* capturadas e distância percorrida por cada câmera utilizando o algoritmo MonoSLAM.

Pode-se observar que o número de *features* capturadas pelas *câmera 2* é relativamente superior ao número de *features* capturadas pela *câmera 1*. Acredita-se que tal diferença ocorre devido à *câmera 2* ter se aproximado do principal e maior obstáculo disposto no ambiente (mesa para oito pessoas). Com isso, a *câmera 2* concentrou uma quantidade maior de *features* na região que abriga o obstáculo mencionado (Figura 3.12).

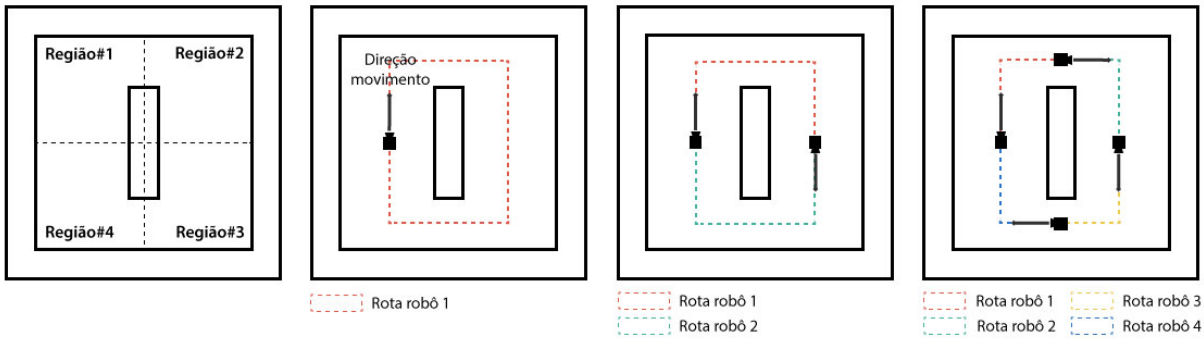


Figura 3.13: Ambiente interno simulado com trajetos pré-definidos.

Considerando que os erros observados podem comprometer a execução de tarefas de exploração, definiu-se, como proposta, estender a execução do algoritmo MonoSLAM utilizando múltiplos agentes. Ao observar que os erros de estimativa acumulam-se durante o tempo de execução do algoritmo, criou-se a hipótese de que, com um número maior de agentes explorando um mesmo ambiente em locais distintos, pode-se diminuir o acúmulo de erros. Isso ocorre devido ao fato de que um ambiente pode ser explorado em um tempo menor através de estratégias de colaboração.

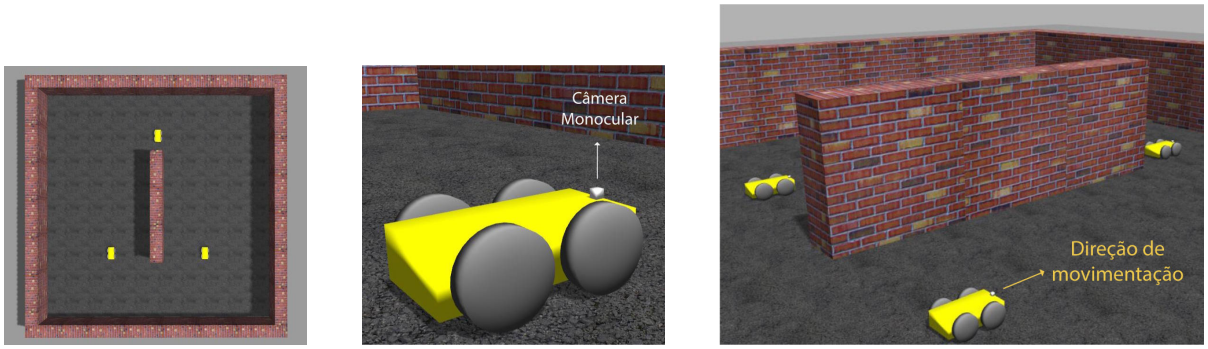
A fim de corroborar tal hipótese, novos experimentos foram definidos, para que fossem confrontadas as abordagens de exploração utilizando um único agente e utilizando múltiplos robôs. Os experimentos mencionados consistem na exploração de um ambiente interno controlado, considerando trajetos pré-definidos, conforme observado na Figura 3.13.

Tais experimentos foram realizados utilizando a ferramenta de simulação robótica Gazebo[3] em conjunto com o ROS. O Gazebo é um simulador de robôs que provém um ambiente no qual algoritmos podem ser testados rapidamente, robôs podem ser projetados de forma fácil e cenários realistas podem ser construídos e testados. O simulador oferece a habilidade de simular eficientemente populações de robôs em ambientes internos complexos e ambientes externos diversos.

O ambiente ilustrado na Figura 3.13 foi projetado utilizando o simulador Gazebo, assim como os robôs utilizados nos experimentos simulados. O ambiente projetado e os robôs são detalhados na Figura 3.14. O robô utilizado

[3]http://gazebosim.org

nas simulações foi projetado sobre uma base sólida, com 4 rodas para movimentação e uma câmera monocular. Nenhum outro sensor foi adicionado à descrição do robô, uma vez que esta dissertação visa explorar o campo de visão pura como único sensor para estimativa de posicionamento dos robôs.



(a) (b) (c)

Figura 3.14: Ambiente e robôs simulados no simulador Gazebo.

A Figura 3.15 demonstra a captura de *features* utilizando o algoritmo MonoSLAM implementado no ROS. Neste experimento, o algoritmo MonoSLAM foi executado com o ambiente e a câmera ilustrados na Figura 3.14. Os resultados dispostos na Figura 3.15 foram obtidos a partir da execução de uma tarefa de exploração utilizando apenas um único robô. A tarefa de exploração em questão não foi realizada de forma autônoma, mas este não é o objetivo do algoritmo MonoSLAM. Para tal, definiu-se uma trajetória específica para que o robô a seguisse.

Considerando o cenário de experimentação descrito na Figura 3.13, os seguintes experimentos foram realizados utilizando o algoritmo MonoSLAM em percursos pré-definidos:

- **Experimento 1**: Percorrer a rota estipulada para um único robô utilizando o algoritmo MonoSLAM;
- **Experimento 2**: Percorrer as rotas estipuladas para dois robôs utilizando o algoritmo MonoSLAM;
- **Experimento 3**: Percorrer as rotas estipuladas para quatro robôs utilizando o algoritmo MonoSLAM;

A Figura 3.16 ilustra os resultados obtidos na execução do primeiro experimento descrito, que utiliza apenas um único robô para percorrer o ambiente.

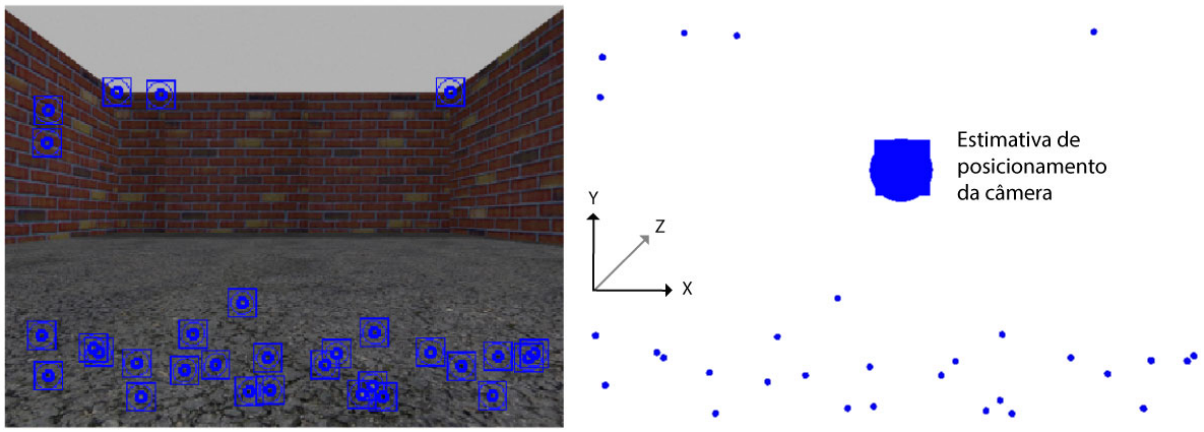


Figura 3.15: Captura de *features* com MonoSLAM considerando um ambiente e uma câmera simulada no simulador Gazebo.

Pode-se notar que o trajeto estimado pelo algoritmo MonoSLAM (segmento verde na Figura 3.16(a)) é consideravelmente próximo ao trajeto planejado (segmento tracejado na Figura 3.16(b)). Os desníveis do terreno e os *frames* de imagem com borrões (oriundos da movimentação acelerada da câmera) são fatores que influenciam diretamente no acúmulo de erros de estimativa durante a execução do trajeto.

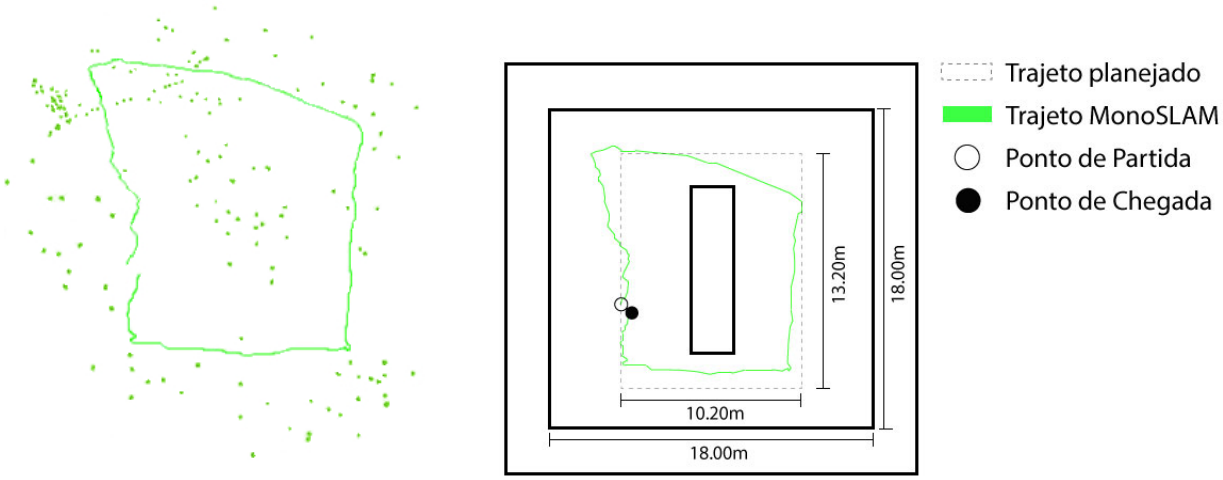
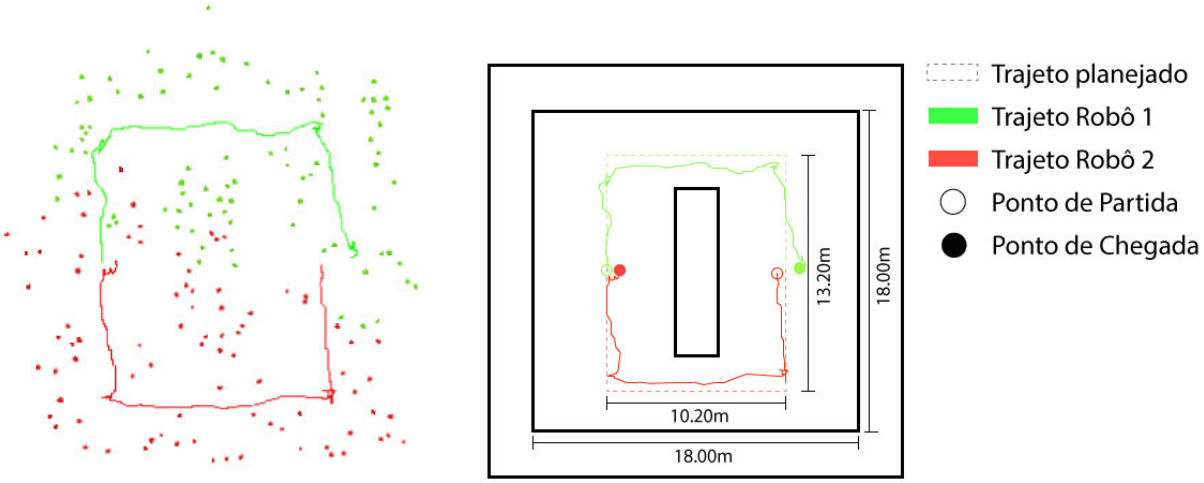


Figura 3.16: Resultados gerados durante a execução do Experimento 1.

A Figura 3.17 ilustra os resultados do segundo experimento realizado, que consiste na exploração de trajetos pré-definidos utilizando 2 robôs. Os resultados obtidos neste experimento mostram que o trajeto estimado pelo algoritmo

MonoSLAM é próximo ao trajeto pré-determinado para cada um dos robôs. Nota-se, ainda, que as trajetórias estimadas pelo algoritmo apresentam, visualmente, erros menores quando comparados com a estimativa ilustrada na Figura 3.16.



(a) (b)

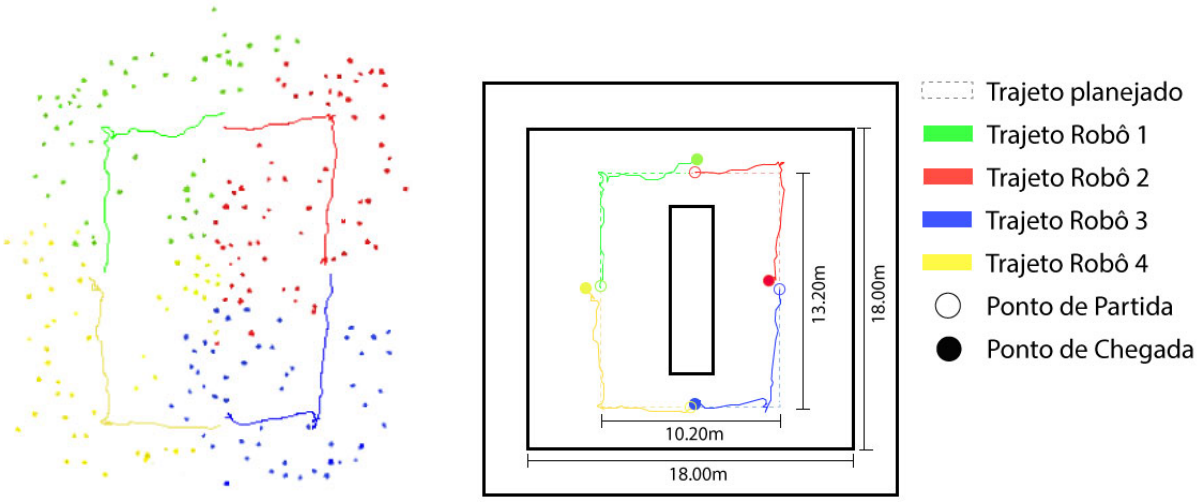
Figura 3.17: Resultados gerados durante a execução do Experimento 2.

O último experimento que envolve a execução do algoritmo MonoSLAM é ilustrado na Figura 3.18. Neste experimento, 4 robôs foram utilizados para explorar o ambiente através das rotas pré-determinadas. Pode-se observar que o trajeto esstimado por cada robô apresenta menores erros em relação aos demais experimentos realizados.

Os dados referentes aos 3 experimentos descritos anteriormente estão dispostos na Tabela 3.2. Nesta, pode-se observar, para cada experimento realizado, a quantidade de *features* capturadas no decorrer do tempo de execução, uma relação de distâncias entre o trajeto planejado e trajeto estimado pelo algoritmo MonoSLAM e o tempo de execução da tarefa de exploração do ambiente. Nos 3 experimentos, a velocidade linear média considerada para cada robô é de $0.5m/s$.

3.5 Considerações finais

Analisando os resultados obtidos, identificou-se que o algoritmo MonoSLAM é suficientemente adequado para prover odometria visual no escopo de aplicação considerado. Esse escopo consiste em ambientes internos controla-



(a) (b)

Figura 3.18: Resultados gerados durante a execução do Experimento 3.

Experimentos		**Quant. de *features* capturadas**	**Dist. percorrida(m)/ Dist. trajeto (m)**	Tempo(m:s)
Exp. 1	Robô 1	157	45.26/46.80	5:05
Exp. 2	Robô 1	90	25.20/26.40	2:39
	Robô 2	86	25.90/26.40	2:43
Exp.3	Robô 1	57	12.86/11.70	1:43
	Robô 2	68	12.08/11.70	1:33
	Robô 3	53	11.90/11.70	1:29
	Robô 4	59	12.68/11.70	1:41

Tabela 3.2: Dados obtidos nos experimentos realizados utilizando o algoritmo MonoSLAM.

dos. Tal limitação se deve ao fato do algoritmo MonoSLAM apresentar complexidade algorítmica $O(N^2)$, onde N é o número de *features* mantidas.

Nos experimentos realizados com a abordagem multi-agente o algoritmo também se mostrou satisfatório, mesmo considerando que todas as instâncias foram executadas em uma máquina central. Os experimentos foram realizados considerando um limite de 100 *features* mantidas por instância de execução. Para fins de visualização manteve-se, para cada instância do MonoSLAM, um tópico alternativo no ROS com as *features* removidas do mapa.

Com relação às estimativas de posicionamento fornecidas pelo algoritmo aplicado, conclui-se que estas são mais fidedignas quando um número maior de robôs é utilizado para a realização de tarefas de exploração. Nesse sen-

tido, pode-se assumir que a utilização de uma abordagem colaborativa para a exploração de um ambiente desconhecido contribui para a diminuição de erro acumulado na estimativa de posicionamento dos agentes, uma vez que o perímetro de exploração é dividido entre os robôs. Com isso, uma distância menor é percorrida por cada robô, fazendo com que o erro de estimativa acumulado seja menor quando comparado com abordagens que consideram um único robô para realizar a mesma tarefa.

No capítulo seguinte, são apresentadas as técnicas bio-inspiradas consideradas na construção da abordagem apresentada na presente dissertação.

CAPÍTULO

4

Otimização por colônia de formigas

4.1 Considerações iniciais

Novos paradigmas e abordagens computacionais vem sendo desenvolvidos com base em processos e mecanismos biológicos observados na natureza (Calvo, 2012). Segundo de Castro (2006), a computação inspirada na natureza é um dos principais ramos da Computação Natural. Este ramo abrange mecanismos desenvolvidos com base em comportamentos e fenômenos observados na natureza.

Os comportamentos biológicos podem ser modelados computacionalmente visando a solução e otimização de processos. Algumas areas de atuação principais podem ser citadas, como algoritmos evolutivos, redes neurais artificiais, sistemas imunológicos artificiais e inteligência de enxames.

O presente trabalho está inserido no ramo de pesquisa discutido (computação inspirada pela natureza), pois a abordagem proposta é baseada na técnica denominada Sistema de Vigilância baseada na Modificação do Sistema Colônias de Formigas (*Inverse Ant System-Based Surveillance System* - IAS-SS) que é, por sua vez, baseada na abordagem de Otimização por Colônia de Formigas (*Ant Colony Optimization* - ACO) proposta por Dorigo (1992). As técnicas IAS-SS e ACO integram o ramo de pesquisa denominado inteligência de enxames.

Segundo de Castro (2006), o ramo de inteligência de enxames baseia-se no

uso de agentes que cooperam entre si e apresentam um grau de inteligência dentro do comportamento social. de Castro (2006) ainda complementa que os agentes, além de cooperarem entre si, devem ser capazes de interagir com o ambiente explorado. Tal interação é análoga ao comportamento biológico observado em algumas espécies de animais. São exemplos de comportamento animal importantes: a busca por comida em um ambiente; a divisão de trabalhos e execução coletiva de tarefas diversas; e a fuga de predadores.

Este capítulo é organizado como segue. Na Seção 4.2, apresenta-se uma série de conceitos teóricos e a formulação do problema. O algoritmo de Colônia de Formigas proposto por Dorigo (1992) é apresentado na Seção 4.3. Por fim, na Seção 4.4, apresenta-se a abordagem IAS-SS proposta por Calvo et al. (2011).

4.2 Fundamentos teóricos

Segundo Dorigo e Stützle (2003), a técnica ACO é uma meta-heurística proposta para a resolução de problemas de otimização combinatória difíceis. Os autores relatam que a fonte de inspiração para a criação do ACO é o comportamento real de colônia de formigas biológicas que utilizam rastros de feromônio como meio de comunicação.

O comportamento biológico observado em colônias de formigas consiste na utilização de trilhas de feromônio que atraem as formigas para um caminho mais curto entre o ninho e a fonte de alimento. A Figura 4.1 ilustra este comportamento.

Formigas biológicas não utilizam-se da visão para encontrar seus objetivos. Em contraproposta, depositam uma substância atrativa (feromônio) no caminho percorrido enquanto buscam um objetivo específico. A substância depositada pode ser detectada pelas demais formigas e uma colônia. Dessa forma, alta concentração de feromônio indica trajetos com maior probabilidade de escolha. Quando uma formiga escolhe um trajeto específico, reforça-se a concentração de feromônio neste trajeto.

De forma análoga ao comportamento biológico, a técnica ACO é baseada na comunicação indireta de uma colônia composta por simples agentes, cha-

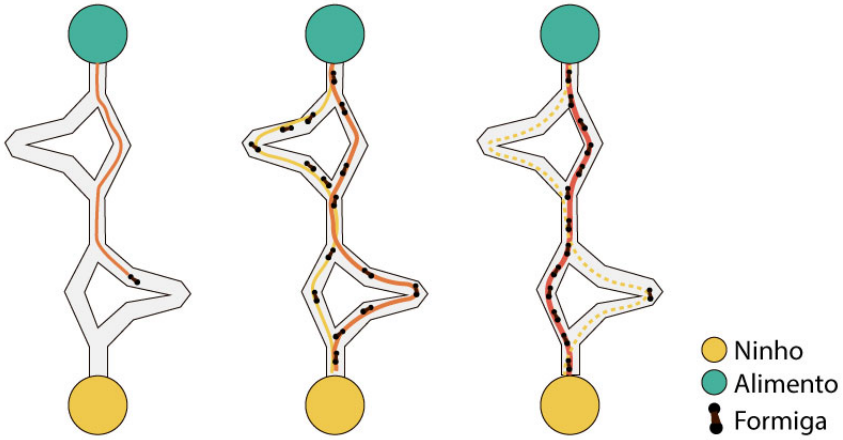


Figura 4.1: Formigas biológicas em busca pelo alimento.

madas por Dorigo e Stützle (2003) de formigas artificiais que criam rastros artificiais de feromônio.

Dorigo e Stützle (2003) definem os rastros de feromônio como sendo informações numéricas distribuídas que são utilizadas pelas formigas artificiais para construir soluções probabilísticas para um problema que deve ser resolvido. As formigas artificiais se adaptam durante a execução do algoritmo visando refletir suas experiências de busca.

O algoritmo *Ant System* (AS) descrito em Maniezzo et al. (1991) e Dorigo (1992) foi o primeiro exemplo baseado no comportamento de colônia de formigas. Na ocasião, o algoritmo AS foi aplicado ao Problema do Caixeiro Viajante (*Traveling Salesman Problem* - TSP) (Johnson e Pilcher, 1988; Reinelt, 1994). Apesar dos resultados encorajadores, Dorigo e Stützle (2003) relatam que o algoritmo AS não pode competir com outros algoritmos do estado da arte propostos para o problema TSP.

Contudo, eles afirmam que o algoritmo AS teve um importante papel de estimulo a outras pesquisas relacionadas para a criação de variantes do algoritmo capazes de obter melhor desempenho computacional. Nesse sentido, pode-se citar aplicações afins que obtiveram bons resultados, como o roteamento de veículos, o roteamento em redes de internet, ordenação sequencial e diversas outras aplicações (Di Caro e Dorigo, 1998; Stützle e Dorigo, 1999; Gambardella et al., 1999; Gambardella e Dorigo, 2000; den Besten et al., 2000; Calvo et al., 2011).

O domínio de aplicação dos algoritmos mencionados é vasto. O algoritmo ACO pode ser aplicado a qualquer problema de otimização discreta para os

quais algum mecanismo de solução pode ser concebido (Dorigo e Stützle, 2003).

4.3 Algoritmo de colônia de formigas

Visando o desenvolvimento de um algoritmo para otimização combinatorial utilizando ACO, considere o conjunto $F = \{f_1, \ldots, f_n\}$, onde $n \in N$ é o conjunto de formigas e um grafo $G(V,C)$, no qual $V = \{v_n, v_a\}$ é o conjunto de nós e $C = \{c_1, c_2\}$ é o conjunto de arestas de G.

O ninho de formigas é representado pelo nó v_n e o local do alimento pelo nó v_a. Os nós v_n e v_a são conectados pelas arestas c_1 e $c_2 \in C$. Associa-se, a cada uma destas arestas, um valor $t_i, i = 1,2$. O valor t_i indica o tamanho do caminho, onde $t_1 > t_2$. Com isso, tem-se que o caminho c_2 apresenta o trajeto mais longo entre o ninho da colônia e a fonte de alimento.

Todavia, as formigas não têm conhecimento destas informações. A concentração de feromônio τ_{ci} no caminho c_i é discretizada. O valor de τ_{ci} é incrementado em uma unidade sempre que o caminho c_i for escolhido por uma formiga. Tem-se, como probabilidade de uma formiga f escolher um caminho c_i:

$$P_{ci} = \frac{\tau_{c_i}}{\sum_{i=1}^{m} \tau_{c_i}} \tag{4.1}$$

onde m é o número de arestas decorrentes do nó no qual a formiga se encontra.

Um outro comportamento biológico observado em colônia de formigas é a evaporação da substância (feromônio) depositada pelas formigas. Com o passar do tempo, a concentração de feromônio em um trajeto tende a reduzir. Após um período, a concentração pode ser nula se o caminho não for mais escolhido por nenhuma formiga.

Analogamente, a concentração de feromônio em um caminho c_i reduz à medida que um número menor de formigas o escolhe. Com isso, o comportamento de evaporação converge à uma solução sub-ótima, uma vez que a exploração aleatória de novos caminhos torna-se mais frequente.

A evaporação em um tempo discreto t pode ser matematicamente modelada como:

$$\tau_{c_i}(t+1) = (1-\rho)\tau_{c_i}(t) \tag{4.2}$$

onde $\rho \in [0,1]$ é a taxa de evaporação.

Considerando os comportamentos descritos, as formigas que estão posicionadas no ninho v_n iniciam uma busca aleatória por alimento. É importante mencionar que a busca é aleatória apenas na primeira iteração. Como nenhum dos caminhos apresenta concentração prévia de feromônio, os caminhos c_1 e c_2 apresentam uma mesma probabilidade de escolha. A Figura 4.2 ilustra o problema exemplo que será considerado no decorrer desta seção.

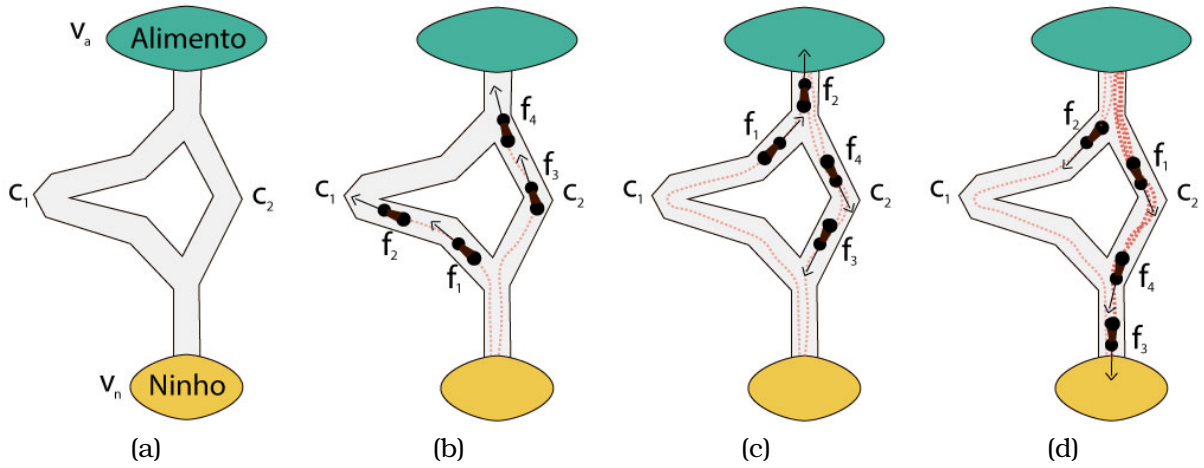


Figura 4.2: Processo de busca pelo caminho ótimo entre o ninho e a fonte de alimento.

Tendo como base o número de formigas $n = 4$, assume-se que as formigas f_1 e f_2 escolhem o caminho c_1 e as formigas f_3 e f_4 escolhem o caminho c_2. Considerando que $t_1 > t_2$, as formigas f_3 e f_4 chegam primeiro no alimento v_a. Com isso, $\tau_{c_2} = 2$. Quando as formigas f_3 e f_4 retornam ao ninho, o caminho c_2 apresentam maior probabilidade de escolha, dado que $\tau_{c_2} > \tau_{c_1}$. Assim, $\tau_{c_2} = 4$.

No instante em que as formigas f_1 e f_2 alcançam a fonte de alimento v_a, tem-se $\tau_{c_1} = 2$. Ao retornarem para o ninho, a probabilidade de escolha de c_2 ainda é maior. Nesse sentido, a tendência é que o feromônio depositado no caminho c_1 desapareça.

Com isso, a concentração de feromônio em c_2 aumenta. Consequentemente, o caminho c_1 será raramente escolhido nas próximas buscas por alimento. Segundo Deneubourg et al. (1990), a maior concentração de feromônios em um caminho c_i evidencia o melhor caminho até a solução do problema.

4.4 Sistema de vigilância baseada na modificação do sistema colônias de formigas

Tendo como base a abordagem de Colônia de Formigas proposta por Dorigo (1992), Calvo et al. (2011) propõe a estratégia de coordenação denominada Sistema de Vigilância baseada na Modificação do Sistema Colônias de Formigas (*Inverse Ant System-Based Surveillance System* - IAS-SS).

No IAS-SS, cada robô move-se de forma independente e toma decisões com base nos estímulos que recebe do ambiente explorado. O IAS-SS propõe uma abordagem na qual cada agente apresenta um comportamento repulsivo à substância que estes depositam durante a movimentação na busca por um objetivo específico. Tal comportamento é inverso ao comportamento tradicional apresentado por Dorigo (1992).

A estrutura de um robô considerada no IAS-SS, assim como nas abordagens tradicionais de formigas biológicas, baseia-se em um princípio de tomada de decisões que considera as substâncias (feromônio) detectadas para que uma direção de movimento seja tomada. Para tal, os robôs são compostos por um controlador de navegação e um módulo de dispersão de feromônio.

O controlador de navegação utiliza-se da substância (feromônio) detectada no ambiente para que o comportamento repulsivo seja gerado. Dessa forma, a direção de movimentação do robô deve condizer com uma região que apresentam baixa concentração de feromônio. Após o processo de detecção de feromônio, o robô deve depositar feromônio considerando uma área de cobertura e alcance de dispersão.

A substância depositada no decorrer de uma tarefa de exploração se estende pelas áreas que são visitadas pelo robô. Com isso, um robô é capaz de criar uma trilha de substância repulsiva a partir de sua posição no ambiente explorado. Tem-se, como objetivo, que a trilha repulsiva seja evitada pelos demais robôs que cumprem uma mesma tarefa. A Figura 4.3 ilustra a arquitetura do sistema cibernético definida para um único robô.

Em abordagens tradicionais como a de Han et al. (2011), o feromônio é depositado apenas na posição corrente de um robô, indicando o caminho percorrido por este. Entretanto, no IAS-SS, o feromônio é depositado além da posição do robô, conforme ilustrado na Figura 4.4. Assim, um robô não precisa se deslocar por todo o ambiente para obter informações.

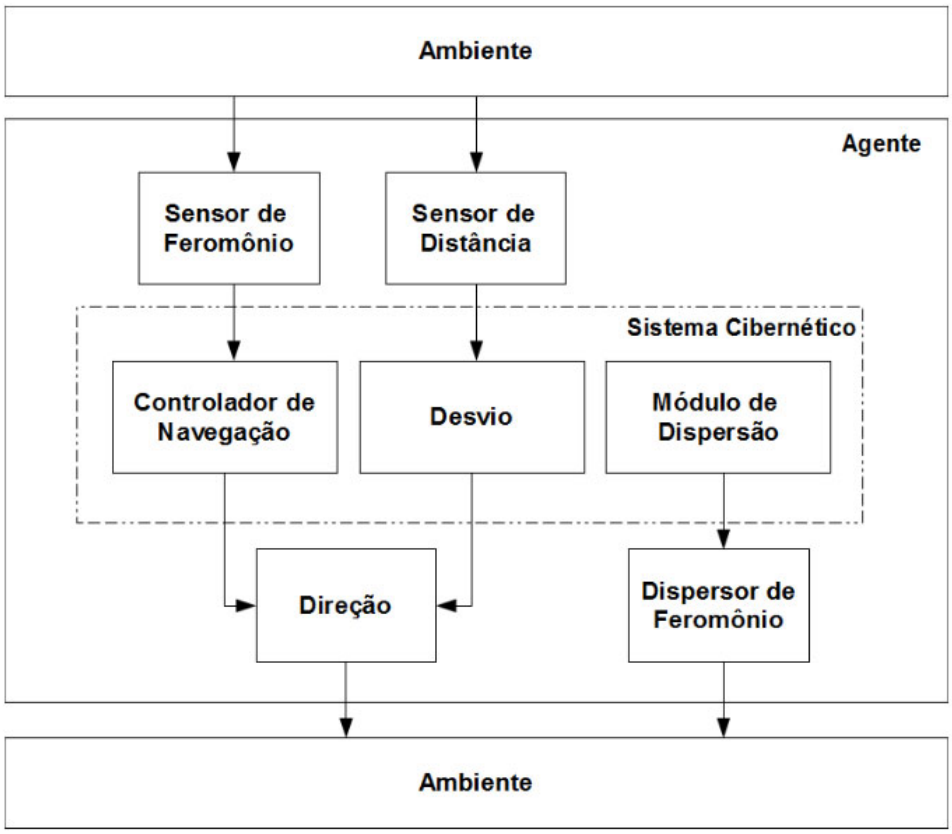


Figura 4.3: Arquitetura do sistema cibernético para um único robô na abordagem IAS-SS. Extraída de: Calvo (2012)

A coordenação dos robôs no IAS-SS é baseada no conceito de estigmergia, um mecanismo de comunicação indireta que é observado em colônias de formigas biológicas. Durante a navegação em uma tarefa de exploração de um ambiente desconhecido, os robôs depositam feromônio no ambiente. O feromônio é análogo ao feromônio utilizado por formigas biológicas.

Por meio de sensores, um robô é capaz de detectar feromônio depositado no ambiente. Antes de continuar a tarefa de exploração, o robô ajusta sua direção de movimento e deposita uma quantidade específica de feromônio. Após um período de tempo, a quantidade de feromônio acumulado nesta área tende a aumentar, visto que outros robôs podem ter visitado a mesma área.

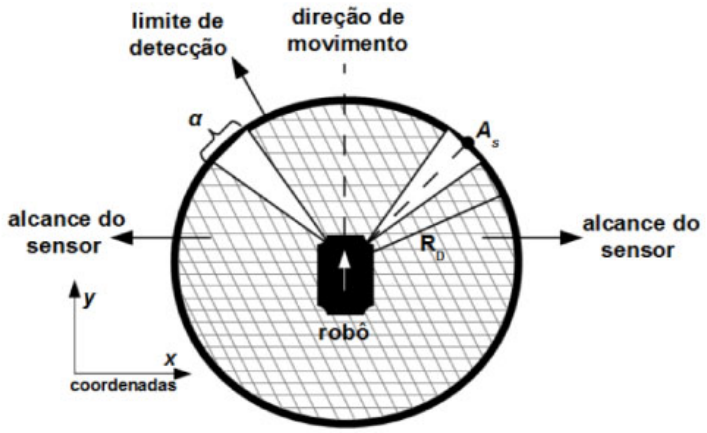


Figura 4.4: Liberação de feromônio no perímetro do robô na abordagem IAS-SS. Retirado de Calvo (2012).

4.5 Considerações finais

Considerando o comportamento de repulsão abordado no IAS-SS, a presente dissertação apresenta um sistema bio-inspirado para a coordenação de múltiplos robôs que utiliza apenas do campo de visão pura como sensor para a estimativa de posicionamento dos robôs. É importante salientar que a abordagem proposta por Calvo et al. (2011) utiliza-se de sensores de distância específicos para prover dados de odometria.

No capítulo a seguir, apresenta-se um sistema bio-inspirado, baseado na abordagem IAS-SS, que tem como objetivo coordenar múltiplos robôs em tarefas de exploração e vigilância de ambientes desconhecidos.

CAPÍTULO

5

Sistema bioinspirado para a coordenação de múltiplos robôs

5.1 Considerações iniciais

De acordo com U. Lima e M. Custódio (2005), robôs móveis são empregados em diversas aplicações potenciais em áreas estratégicas como vigilância de ambientes, sistemas de transporte, detecção de incêndio florestal, busca e resgate em desastres de grande escala e redes de sensores autônomos. Aulinas et al. (2008) acrescentam que nestas tarefas, robôs são posicionados em um ambiente conhecido ou desconhecido e devem ser capazes de construir, simultaneamente, um modelo (mapa) do ambiente explorado e determinar suas próprias posições dentro do ambiente explorado.

A presente dissertação propõe um sistema bio-inspirado, denominado PheroSLAM, que utiliza uma versão modificada da abordagem de Colônia de Formigas para coordenar múltiplos robôs em tarefas de exploração e vigilância de ambientes desconhecidos. O PheroSLAM consiste, basicamente, em utilizar trilhas de feromônio com característica repulsiva para guiar múltiplos robôs.

No PheroSLAM, cada robô deve navegar e construir um modelo (mapa) de um ambiente desconhecido ou parcialmente desconhecido. Para tal, um robô deve conhecer sua própria posição dentro do ambiente explorado. Dessa

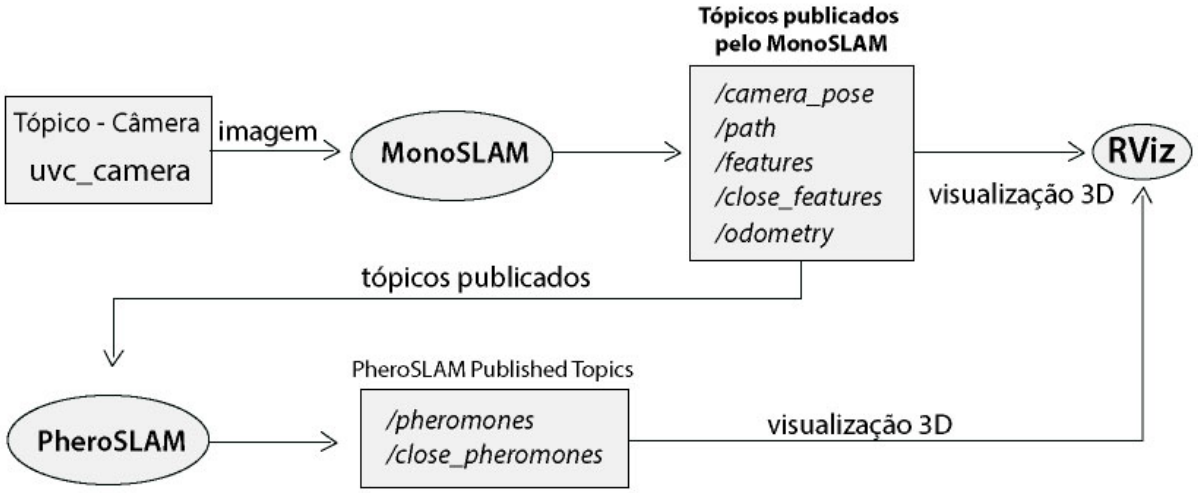


Figura 5.1: Grafo da arquitetura definida para o PheroSLAM no ROS.

forma, visando prover dados de odometria para que cada robôs consiga se localizar no ambiente, propõe-se a utilização do algoritmo MonoSLAM proposto por Davison et al. (2007) e discutido no Capítulo 3.

Assim como o MonoSLAM, o sistema PheroSLAM foi implementado sobre o ROS (Robot Operating System) utilizando a linguagem `C++` e o sistema operacional `Ubuntu Linux 14.04`. Todos os dados necessários para a execução do sistema, como mapas probabilísticos, odometria visual e estimativas de posicionamento, são providos como tópicos no ROS. A Figura 5.1 ilustra a arquitetura de implementação definida para o PheroSLAM. Tal arquitetura também inclui os tópicos publicados pelo nó de execução do algoritmo MonoSLAM.

Os dados de odometria visual necessários para que o PheroSLAM possa guiar e coordenar múltiplos robôs em tarefas de exploração em vigilância de ambiente são fornecidos como tópicos pelo nó de execução do MonoSLAM no ROS. Na arquitetura proposta (Figura 5.1), os tópicos publicados pelo MonoSLAM são responsáveis por prover: a estimativa de posicionamento da câmera do robô (`\camera_pose`), a estimativa de caminho percorrido pelo robô no decorrer de tempo de exploração (`\path`), um mapa tridimensional probabilístico contendo todas as *features* capturadas no decorrer da tarefa (`\features`), um mapa tridimensional probabilístico contendo apenas as *features* próximas à câmera do robô (`\close_features`) e um quaternion com dados de odometria visual (`\odometry`). Os dados referentes a cada tópico podem ser visualizados na ferramenta ROS RViz.

Este capítulo é organizado como segue. Na Seção 5.2, apresenta-se uma série de conceitos teóricos e a formulação do problema (PheroSLAM). Na Seção 5.3, detalhes de implementação são apresentados. O sistema de navegação bio-inspirado proposto é detalhado na Seção 5.4. Os experimentos e resultados obtidos são discutidos na Seção 5.5. Por fim, na Seção 5.6, as considerações finais deste capítulo são fornecidas.

5.2 Formulação do problema

A coordenação de múltiplos robôs tem um papel importante no campo de robótica colaborativa e em aplicações de exploração cooperativa utilizando múltiplos robôs (Sheng et al., 2006). Considerando o problema de explorar ambientes desconhecidos ou parcialmente desconhecidos utilizando múltiplos robôs, é desejável que cada robô tenha como objetivo ótimo visitar áreas ainda não exploradas. Entretanto, coordenar um time de robôs a fim de atingir o objetivo em questão é uma tarefa complexa.

Para tal, uma estratégia eficiente de coordenação de múltiplos robôs é necessária, uma vez que cada robô deve ser capaz de determinar seu posicionamento com relação aos demais robôs que executam a mesma tarefa. Em tarefas de exploração coletiva, os robôs devem explorar cada área desconhecida de forma coordenada, até que o ambiente seja completamente visitado. Existem diversas abordagens propostas para resolver o problema supracitado. Dorigo (1992) afirma que a utilização de algoritmos bio-inspirados é uma das principais abordagens evidenciadas na literatura.

Os dados de odometria visual necessários para o funcionamento efetivo do sistema PheroSLAM são providos pelo nó de execução do algoritmo MonoSLAM no ROS. Entretanto, a estimativa de dados odométricos é apenas um dos problemas que deve ser resolvido em tarefas de exploração e vigilância de ambientes. Além das estimativas de posicionamento, um robô deve ser capaz de se mover dentro do ambiente desconhecido buscando por regiões que apresentam maior ganho de informação (regiões ainda não exploradas).

Durante a exploração do ambiente, um mapa esparso de *features* é construído em tempo real através da execução do algoritmo MonoSLAM. Contudo, conforme mencionado anteriormente, uma abordagem efetiva para a coordenação de múltiplos robôs é necessária. Nesse sentido, a abordagem proposta para o sistema PheroSLAM considera um mecanismo bio-inspirado que define

uma organização social de sistemas coletivos. Tal mecanismo é baseado na abordagem *Ant System-Based Surveillance System* (IAS-SS) apresentada por Calvo et al. (2011).

Na abordagem IAS-SS, os robôs depositam feromônio em um ambiente desconhecido enquanto o exploram. Entretanto, a estratégia adotada no IAS-SS é oposta às estratégias tradicionais propostas em outros sistemas baseados na abordagem de Colônias de Formigas. Abordagens tradicionais baseadas em Colônias de Formigas, como a proposta de Dorigo (1992), liberam uma quantidade de feromônio no ambiente explorado em posições específicas, sinalizando a trajetória percorrida pelos robôs. Tais trajetórias apresentam características de atração para os robôs, uma vez que podem indicar uma possível rota para a resolução de um problema.

Em contraproposta, na abordagem IAS-SS os robôs depositam substâncias (feromônio) repulsivas. Tem-se, como princípio, que cada robô que executa tarefas de exploração e vigilância de ambiente deve evitar áreas recentemente exploradas, a fim de maximizar a cobertura e vigilância do ambiente em questão. Dessa forma, cada robô evita regiões com altas concentrações de feromônio, uma vez que tal concentração indica regiões recentemente visitadas. No IAS-SS, diferentemente das abordagens tradicionais, deposita-se feromônio em uma ampla área à frente do robô considerando um limite de alcance do sensor de detecção de feromônio.

As abordagens propostas por Calvo et al. (2011) e Dorigo (1992) podem ser aplicadas apenas com agentes terrestres, em um espaço de exploração bidimensional. Dessa forma, as abordagens supracitadas podem gerar mapas bidimensionais com as concentrações de feromônio no espaço explorado. Com isso, possibilita-se apenas a exploração de ambientes utilizando robôs terrestres.

No entanto, em diversos cenários, outras dimensões são necessárias, dado que um time de robôs pode ser composto por veículos aéreos não-tripulados (VANTs). De acordo com Jiménez Lugo e Zell (2014), a popularidade de pesquisas que envolvem VANTs é crescente nos últimos anos. VANTs popularizaram-se devido a características interessantes como robustez, simplicidade, tamanho reduzido e baixo peso. Trabalhos recentes tem evidenciado tais características, como a execução de manobras arriscadas através de janelas (Mellinger et al., 2014) e monitoramento de longo prazo da floresta amazônica (Pinagé et al., 2013).

A presente dissertação propõe o sistema PheroSLAM como um sistema bio-inspirado que utiliza uma versão modificada da abordagem de Colônia de Formigas, o IAS-SS, para guiar múltiplos robôs em tarefas de exploração e vigilância de ambientes. Além do IAS-SS, o algoritmo MonoSLAM é utilizado para prover dados de odometria visual para os robôs.

Com a utilização do MonoSLAM, prepara-se um ambiente tridimensional para a dispersão de feromônio, uma vez que o MonoSLAM é capaz de manter um mapa probabilístico tridimensional das *features* salientes identificadas no ambiente explorado. Com isso, em trabalhos futuros, o PheroSLAM pode ser estendido para coordenar múltiplos VANTs.

5.3 Detalhes de implementação e execução

A implementação do algoritmo PheroSLAM idealizada na presente dissertação também considera múltiplas instâncias de execução, visto que tem-se como proposta a implementação de um ambiente que fornece recursos para a execução de tarefas de exploração e vigilância de ambientes desconhecidos utilizando múltiplos robôs.

Assim como na execução do algoritmo MonoSLAM, a execução de múltiplas instâncias do PheroSLAM ocorre por meio de arquivos de configuração e diretivas de execução em linha de comando. Para tal, as instâncias são inicializadas e uma identificação é fornecida para cada um dos robôs que compõem uma equipe. A diretiva de execução do nó PheroSLAM pode ser observada a seguir:

```
$ rosrun pheroslam pheroslam conf.xml r1 -x 1 -y 4 -d right
```

Na linha de comando descrita anteriormente, observa-se a execução o nó `pheroslam`. São fornecidos o arquivo de configuração da câmera utilizada no formato `XML`, a identificação do robô (`robot1`), a posição inicial do robô no mapa (`-x 1 -y 4`) e a direção de movimentação inicial do robô (`-d right`).

No momento de inicialização do sistema, o mapa do ambiente explorado passa a ser construído de forma condizente com o posicionamento real dos robôs. A Figura 5.2 ilustra a arquitetura de execução de múltiplas instâncias do nó PheroSLAM definida. Assim como nos experimentos realizados com o algoritmo MonoSLAM, apenas uma máquina é utilizada para executar as instâncias do PheroSLAM.

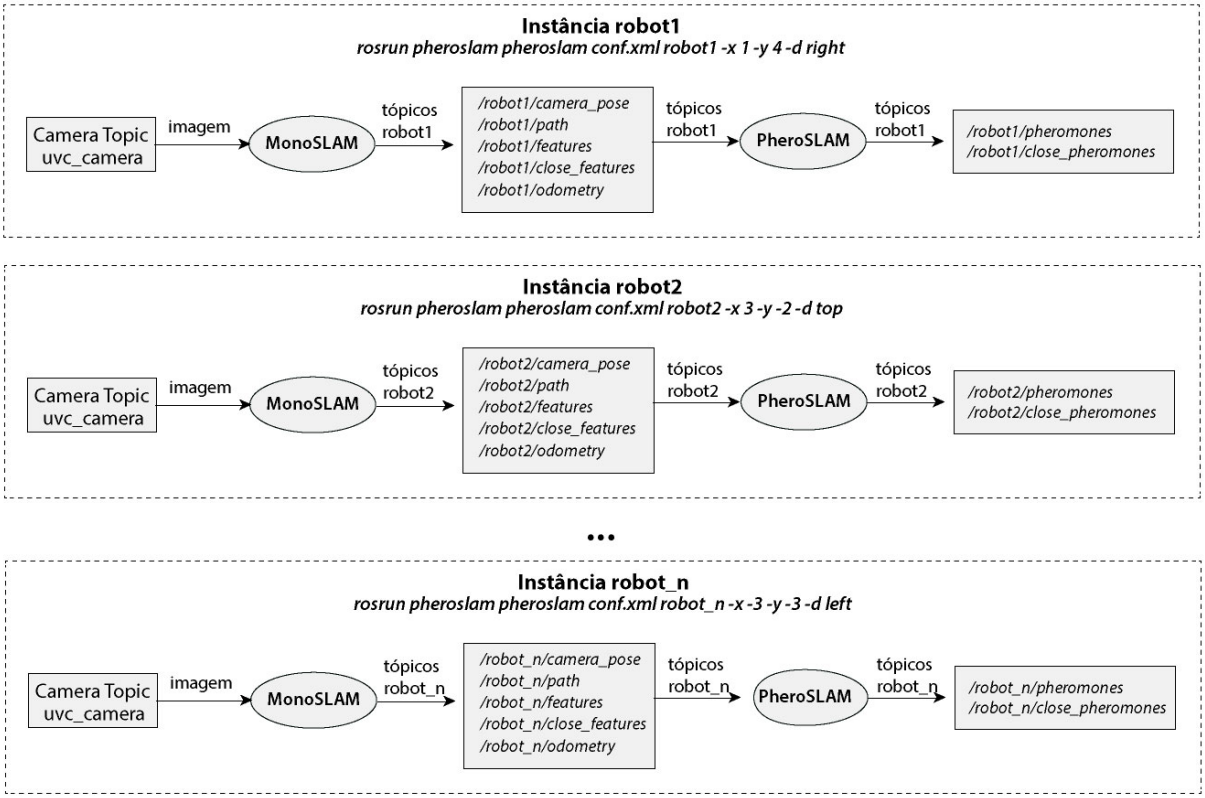


Figura 5.2: Arquitetura definida para a execução de múltiplas instâncias do nó PheroSLAM no ROS.

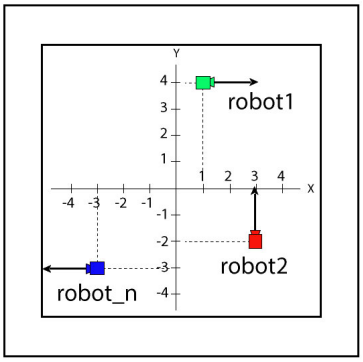


Figura 5.3: Configuração de formação (posição e direção de movimentos iniciais) dos robôs de acordo com a arquitetura ilustrada na Figura 5.2.

Considerando a arquitetura definida anteriormente, observa-se que a imagem capturada por cada instância de execução é transmitida para o nó MonoSLAM, que é responsável basicamente por inferir a estimativa de posicionamento dos robôs e fornecer dados de odometria para o nó PheroSLAM.

O nó PheroSLAM, por sua vez, subscreve os tópicos publicados pelo nó MonoSLAM. Os tópicos subscritos são essenciais para que os robôs possam se movimentar e explorar o ambiente de forma autônoma, evitando minimamente os obstáculos encontrados. A Figura 5.3 ilustra a configuração de formação dos robôs (posição inicial e direção de movimento de cada robô) referente à formação definida na Figura 5.2. A partir dessa formação inicial, o sistema está pronto para que as tarefas de exploração e vigilância sejam realizadas.

5.4 Sistema de navegação e exploração

5.4.1 Mecanismo de depósito de feromônio

Em cada iteração t de execução de uma tarefa de exploração, cada robô deve depositar uma quantidade específica de feromônio ao seu redor, considerando a estimativa de posicionamento deste robô dada pelo algoritmo MonoSLAM. Em termos práticos, a estratégia ilustrada é responsável pela criação de um rastro tridimensional de substâncias (feromônio) repulsivas. O feromônio

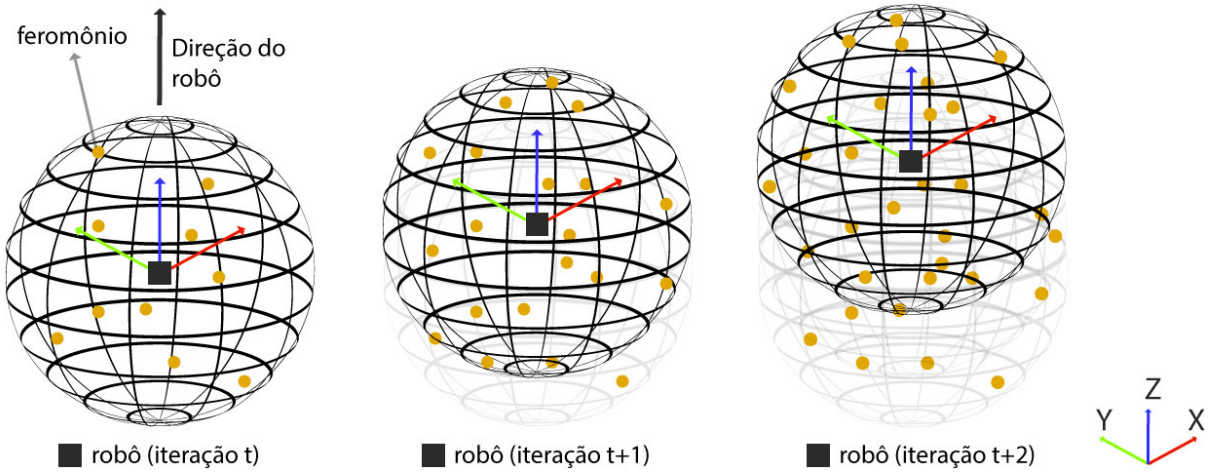


Figura 5.4: Abordagem adotada para o depósito de feromônio.

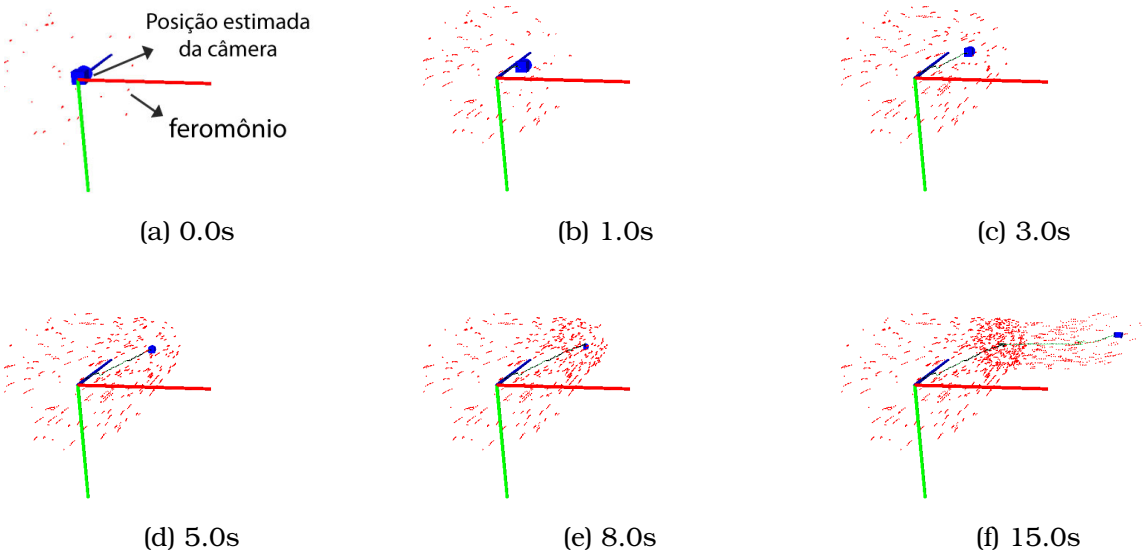
é depositado considerando um limite de alcance esférico ao redor do próprio robô, conforme ilustrado na Figura 5.4.

Diferentemente da abordagem IAS-SS, o feromônio é representado por um ponto 3D em uma nuvem de pontos tridimensional. A nuvem de pontos representa um mapa de feromônios e é disponibilizada como um tópico pelo nó de execução do PheroSLAM no ROS. Dessa forma, através de uma ferramenta como o ROS RViz pode-se visualizar o mapa de feromônios gerado por cada robô.

A Figura 5.5 mostra uma sequência de quadros obtidos na ferramenta RViz, nos quais pode-se observar o mapa tridimensional de feromônio (nuvem de pontos) sendo construído iterativamente. Cada um dos quadros exibidos foi capturado em um instante de tempo específico. O instante de tempo relativo à captura de cada quadro é dado em segundos, conforme observado na mesma figura. Os eixos vermelho, verde e azul representam os eixos X, Y e Z em uma projeção tridimensional, respectivamente. A posição estimada da câmera do veículo é ilustrada pela figura geométrica em destaque na cor azul.

O exemplo ilustrado foi obtido a partir da execução do PheroSLAM no ROS considerando um cenário de experimentação real (não simulado). No experimento em questão utilizou-se uma câmera monocular acoplada a um veículo móvel para prover imagens de uma simples tarefa de exploração.

Para que o mapa de feromônios seja mantido a cada iteração t, o PheroSLAM define um limite esférico para o depósito de feromônio ao redor de um robô. Uma quantidade pré-definida de pontos 3D (coordenadas tridimensio-



(a) 0.0s (b) 1.0s (c) 3.0s

(d) 5.0s (e) 8.0s (f) 15.0s

Figura 5.5: Sequência de quadros ilustrando a construção iterativa do mapa de feromônio (nuvem de pontos).

nais) é aleatoriamente gerada usando uma distribuição normal (Gaussiana). Todos os pontos 3D são gerados dentro do limite imposto pela esfera limitadora definida.

É importante salientar que a estrutura da esfera que limita o alcance do sensor de feromônio é invisível. Tal estrutura é apenas utilizada para delimitar o alcance de dispersão do feromônio depositado ao redor do robô. A quantidade de feromônio depositada a cada iteração t foi empiricamente definida durante a fase de experimentação. Definiu-se que a cada iteração t são gerados 10 pontos 3D, que representam o feromônio gerado nesta iteração. Nas iterações seguintes, novos pontos são gerados e estes passam a compor a nuvem de pontos mantida como mapa de feromônio pelo nó de execução do PheroSLAM.

Considerando que um feromônio não é uma substância estável e evapora de acordo com uma taxa de evaporação específica, com o decorrer da execução das tarefas de exploração e vigilância a concentração de feromônio tende a reduzir. Assim, áreas já exploradas pelos robôs, que foram eventualmente evitadas, podem ser revisitadas. Este comportamento é essencial para a execução de tarefas de vigilância.

No IAS-SS, a quantidade total de feromônio que evapora $\Phi_q(t)$ em uma

posição específica q no instante t é matematicamente modelada por:

$$\Phi_q(t) = \rho\tau_q(t) \tag{5.1}$$

onde ρ, $0 \leq \rho \leq 1$, é a taxa de evaporação e $\tau_q(t)$ é a quantidade total de feromônio na posição q no instante t.

No IAS-SS, o feromônio depositado apresenta uma taxa de concentração na posição q. Essa taxa de concentração pode ser, por exemplo, representada por um valor numérico que varia de 0 a 100, onde zero é a concentração mínima e 100 a concentração máxima de substância.

Como no PheroSLAM cada feromônio é modelado como um ponto 3D em uma nuvem de pontos tridimensional, um ponto 3D no mapa de feromônio mantido indica a presença de feromônio na posição indicada. A concentração de feromônio em uma região específica é dada pela quantidade de pontos (feromônio) identificada na região em questão, diferente do que é proposto no IAS-SS.

O PheroSLAM tem como princípio que cada robô busque regiões com baixas concentrações de feromônio. Considerando que no PheroSLAM a concentração é dada pela contagem de pontos (feromônio) em uma região específica, definiu-se que cada ponto possui um tempo de existência no mapa de feromônios mantido. Dessa forma, a cada iteração, os pontos mais antigos no mapa de feromônios são descartados. Matematicamente, a quantidade total de feromônio $\tau_q(t)$ na posição q no instante t é dada por:

$$\tau_q(t) = (\tau_q(t-1) - \Phi_q(t-1)) + \sum_{i=1}^{N} \Delta_q^k(t) \tag{5.2}$$

onde $\Delta_q^k(t)$ é a quantidade de feromônio depositada pelo k-ésimo robô na posição q no instante t. No PheroSLAM, cada robô gera aleatoriamente 10 pontos 3D (feromônio) ao seu redor no instante t. O limite de alcance do sensor de depósito de feromônio é de 1 metro e foi definido empiricamente durante a fase de experimentação da proposta. Com isso, $\Delta_q^k(t)$ é um valor constante a cada iteração do sistema.

5.4.2 Sistema de navegação

Em abordagens tradicionais de Colônia de Formigas, cada robô deposita uma quantidade específica de feromônio no seu próprio caminho (Dorigo,

1992). Tal ação evidencia o passado de cada robô no ambiente que é explorado. Cada robô cria uma trilha de feromônio que caracteriza seu histórico no decorrer da execução de uma tarefa. A trilha criada pode auxiliar o robô em situações futuras nas tarefas de exploração e de vigilância.

Apesar de existirem diversos tipos de feromônio liberados por formigas, no PheroSLAM cada robô constrói uma trilha artificial repulsiva de feromônio. A ausência desse tipo de substância (feromônio) no ambiente indica áreas com alto potencial para possível monitoramento e exploração. Por outro lado, áreas com alta concentração de feromônio denotam regiões recentemente visitadas. Segundo Calvo et al. (2011), o comportamento repulsivo observado tende a distanciar um robô dos demais, o que beneficia a execução de tarefas de vigilância .

No PheroSLAM, simultaneamente ao processo de dispersão de feromônio, um robô deve buscar por áreas que apresentam baixas concentrações de feromônio. Tem-se como princípio que cada robô deve evitar áreas com alta concentração de feromônio. Em algumas tarefas de exploração, por exemplo, deseja-se que uma área não seja constantemente revisitada. Quando um robô identifica localmente uma área com baixa concentração de feromônio, essa área é considerada como uma potencial região para exploração. Com isso, tal região passa a ser um alvo para o robô.

Tendo como foco encontrar regiões que apresentam baixas concentrações de feromônio, foi implementado um algoritmo, denominado LPCS (*Low Pheromone Concentration Search algorithm*). O LPCS subdivide, recursivamente, uma área em 4 quadrantes de busca. O intuito é encontrar qual dos 4 quadrantes definidos apresenta a menor quantidade de feromônio. O algoritmo termina sua execução quando todos os quatro quadrantes resultantes de uma subdivisão não apresentam concentração de feromônio, conforme ilustrado na Figura 5.6.

Os passos de execução referentes ao exemplo ilustrado na Figura 5.6 são descritos a seguir:

- **Passo 1**: Uma área circular ao redor do robô é definida considerando o raio de alcance do sensor de depósito do feromônio. Como os robôs utilizados nos experimentos são robôs terrestres, a área circular que limita a busca por feromônio é bidimensional, considerando apenas os eixos x e y (não considera a altura dos objetos dispostos no ambiente). A es-

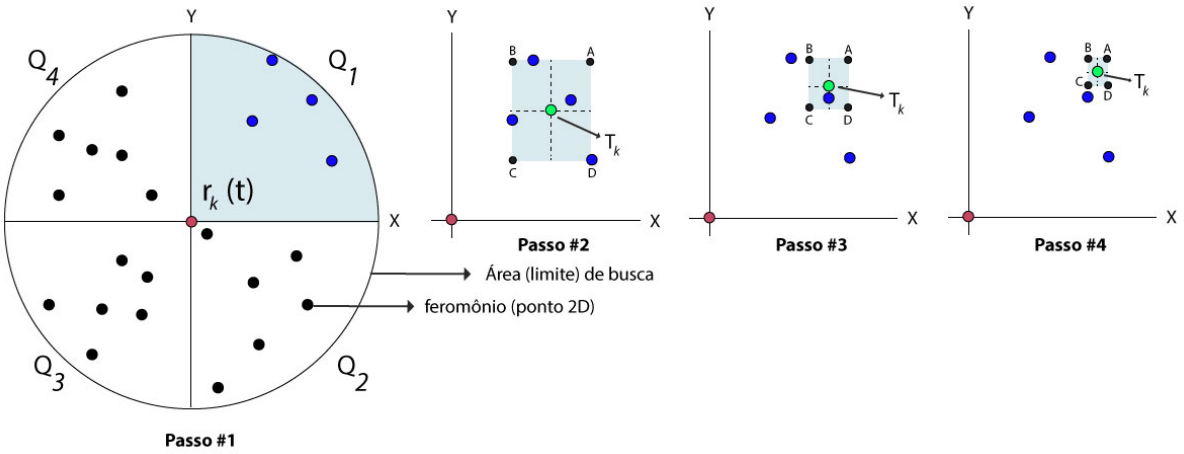


Figura 5.6: Exemplo de execução passo-a-passo do algoritmo LPCS.

timativa de posicionamento do robô $r_k(t)$ em uma iteração t é utilizada como centro da área de busca. O algoritmo LPCS divide essa área inicial em 4 quadrantes (Q_1, Q_2, Q_3 e Q_4) e busca, em cada um destes, aquele que apresenta a menor concentração (quantidade de pontos) de feromônio. A região que apresentar a menor concentração é escolhida como nova região de busca. Na Figura 5.6, o quadrante Q_1 é o que apresenta a menor concentração de feromônio e, portanto, é uma potencial área de exploração. Com isso, Q_1 é considerada como nova região de busca;

- **Passo 2**: Considerando que o quadrante Q_1 é o que apresenta a menor concentração de feromônios dentre os demais, novos quadrantes devem ser recursivamente definidos para que a busca por regiões com baixa concentração de feromônio seja refinada. Dados os pontos bidimensionais que se encontram no novo quadrante de busca (Q_1), cria-se uma nova área de busca, delimitada pelos pontos A, B, C e D. Tais pontos delimitam uma área que agrega todos os pontos encontrados no quadrante Q_1. A área delimitada pelos pontos A, B, C e D passa a ser a nova região de busca considerada pelo algoritmo. Para que a direção do robô seja ajustada e a nova região de busca seja alcançada, um alvo T_k é definido para o robô $r_k(t)$. T_k definido como o ponto central entre os pontos A, B, C e D. No final da execução do algoritmo, o alvo T_k deverá ser perseguido pelo robô $r_k(t)$;

- **Passo 3**: Como os quatro novos quadrantes definidos apresentam a mesma concentração de feromônio escolhe-se, aleatoriamente, um des-

tes para ser a nova região de busca. Na Figura 5.6, o quadrante Q_1 foi selecionado. A partir deste momento, o quadrante Q_1 é a nova região de busca considerada pelo algoritmo. Na área de busca delimitada pelos pontos A, B, C e D, busca-se novamente pelo quadrante que apresenta a menos quantidade de feromônio. Nessa etapa, os quadrante Q_1 e Q_4 apresentam as menores concentrações. Aleatoriamente, o quadrante Q_1 foi selecionado como nova região de busca. Com isso, o alvo T_k também é ajustado de acordo com os novos pontos A, B, C e D encontrados;

- **Passo 4**: O quarto e último passo do exemplo ilustrado na Figura 5.6 mostra a última iteração do algoritmo LPCS. Nesta iteração pode-se observar que os quatro quadrantes Q_1, Q_2, Q_3 e Q_4 não apresentam concentração de feromônios. Aqui, assume-se que a área com menor concentração de feromônios foi encontrada. O alvo T_k é ajustado considerando os novos pontos A, B, C e D e deve ser perseguido pelo robô $r_k(t)$ na iteração $t+1$.

Em síntese, o algoritmo LPCS tem como objetivo buscar a área que apresenta a menor concentração de feromônio considerando como região de busca apenas a área delimitada pelo sensor de detecção de feromônios do robô. Para tal, o algoritmo LPCS subdivide recursivamente a área de busca inicial em quatro quadrantes, buscando sempre encontrar o quadrante com a menor concentração de feromônio. Tem-se como critério de parada encontrar quatro quadrantes que não apresentam concentração de feromônio. Os passos descritos anteriormente podem ser observados no pseudo-código disposto no

Algoritmo 1:

Algoritmo 1: LPCS

entrada: Uma nuvem de pontos $detected_p$, contendo apenas feromônio que está na área de detecção do robô, e um ponto de origem *origin* (estimativa de posicionamento do robô)

saída : Um quaternion *target*, que representa a área com a menor concentração de feromônio encontrada

início

$q_1 \longleftarrow \forall p(x,y,z) \in detected_p$ onde $p.x > origin.x \wedge p.y > origin.y$;

$q_2 \longleftarrow \forall p(x,y,z) \in detected_p$ onde $p.x < origin.x \wedge p.y > origin.y$;

$q_3 \longleftarrow \forall p(x,y,z) \in detected_p$ onde $p.x < origin.x \wedge p.y < origin.y$;

$q_4 \longleftarrow \forall p(x,y,z) \in detected_p$ onde $p.x > origin.x \wedge p.y < origin.y$;

Encontrar o quadrante com o menor número de feromônio;

$lowest \longleftarrow findLowest(q_1, q_2, q_3, q_4)$;

Determinar um ponto central no quadrante que apresenta a menor quantidade de feromônio;

$target \longleftarrow findCenter(lowest)$;

se *numberOfPheromone*(q_1, q_2, q_3, q_4) = 0 **então**

Retornar o quaternion com o alvo encontrado;

retornar *target*;

senão

Chamada recursiva do algoritmo LPCS;

$LPCS(target, lowest)$;

fim

fim

Considera-se que a área na qual se encontra o alvo T_k em uma iteração t é a área que apresenta o maior ganho de informação para o sistema, uma vez que esta área é a que apresenta a menor concentração de feromônio dentro da região de busca proposta. Com esse mecanismo de busca, as regiões com maior concentração causam um efeito repulsivo capaz de redirecionar as trajetórias dos robôs para as regiões que apresentam menores concentrações de feromônio.

Após a execução do LPCS em uma iteração t, o robô $r_k(t)$ deve perseguir o alvo T_k. Para tal, analisa-se o vetor r^W em busca do quaternion q^{RW}, que contém as informações de odometria do robô $r_k(t)$, e os vetores de velocidade angular

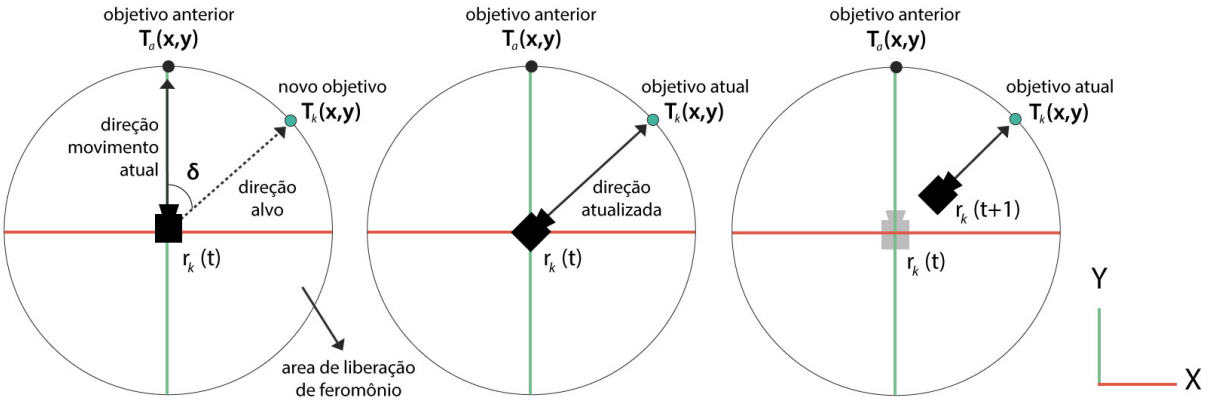


Figura 5.7: Ajuste de direção do robô $r_k(t)$ considerando o objetivo alvo anterior T_a e o novo objetivo T_k.

ω^R e velocidade linear v^w para que a direção do movimento seja ajustada de acordo com a direção dada pelo alvo T_k. A Figura 5.7 ilustra o ajuste de direção δ do robô $r_k(t)$ entre o objetivo anterior T_a e o novo objetivo T_k:

A partir do ajuste de direção do robô $r_k(t)$ ilustrado na Figura 5.7, o sistema direciona o robô para a nova direção de movimentação definida. Com isso, os vetores de velocidade angular ω^R e velocidade linear v^w são atualizados.

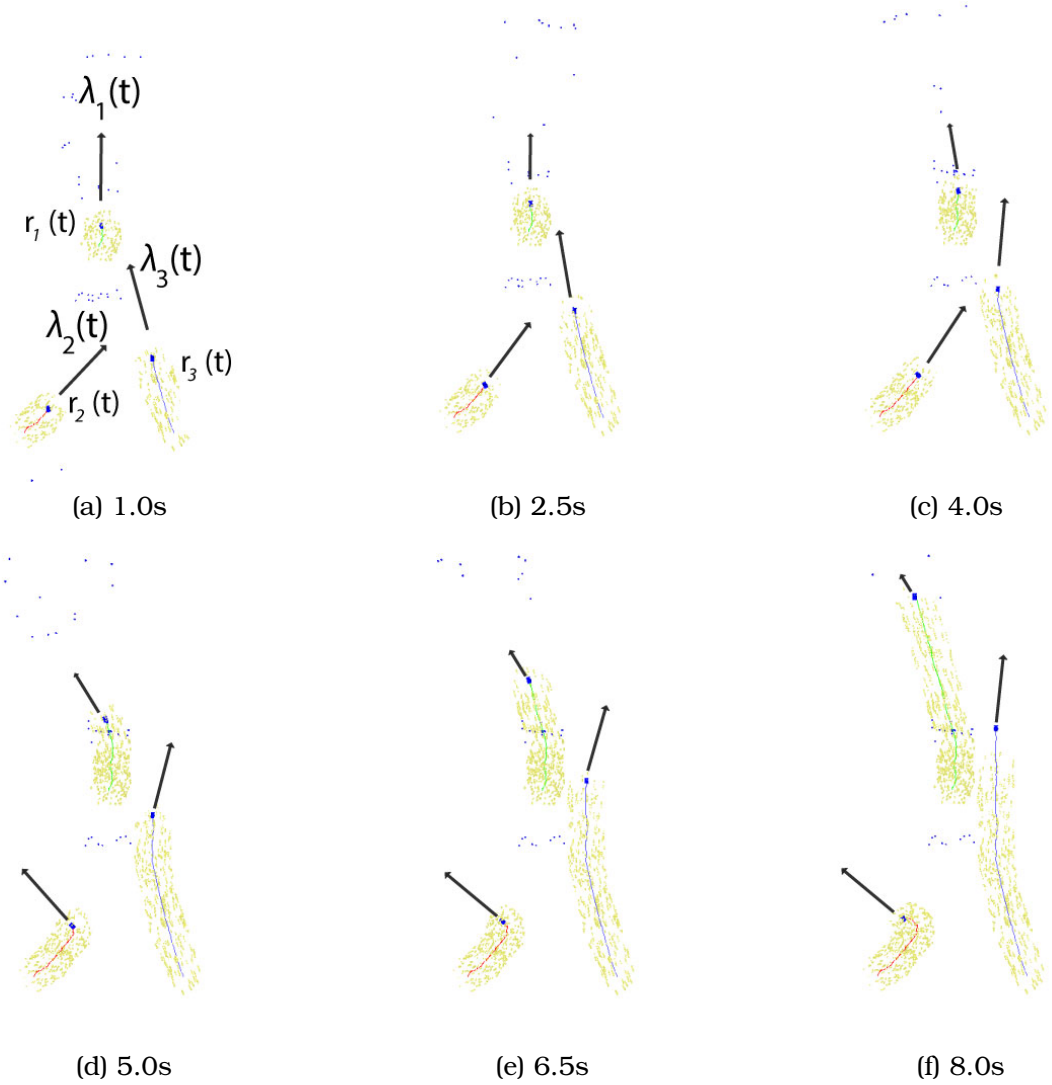
Entretanto, a nova direção de movimento pode ser modificada na próxima iteração do PheroSLAM executada pelo robô $r_k(t)$. Isso devido ao fato que um outro robô $r_l(t)$ que executa uma outra instância do PheroSLAM, pode estar próximo ao robô $r_k(t)$. Se o robô $r_l(t)$ também segue para a mesma região alvo do robô $r_k(t)$, tem-se uma situação de conflito, pois as trilhas de feromônio devem ocasionar um efeito de repulsão.

Dessa forma, a cada nova iteração, os robôs são capazes de atualizar dinamicamente suas direções de movimentação em busca das regiões que acusam maior ganho de informação. Nesse sentido, o alvo estabelecido através do algoritmo LPCS não deve ser obrigatoriamente alcançado. O alvo T_k é utilizado apenas para corrigir a direção de movimento do robô $r_k(t)$.

A Figura 5.8 mostra alguns quadros de execução do PheroSLAM capturados na ferramenta ROS RViz nos quais pode-se observar o comportamento repulsivo gerado pelas trilhas de feromônio (rastro amarelado ao redor do caminho percorrido pelos robôs) durante o passar do tempo (em segundos). Neste experimento, 3 robôs realizam uma tarefa de exploração em um ambiente simulado na ferramenta Gazebo.

Cada um dos robôs $r_1(t)$, $r_2(t)$ e $r_3(t)$ executa uma instância distinta do

PheroSLAM no ROS. Nota-se que, quando os robôs aproximam-se uns dos outros, os sensores de detecção de feromônio identificam regiões com maiores concentrações de feromônios por meio do algoritmo LPCS e as direções de movimentação $\lambda_1(t)$, $\lambda_2(t)$ e $\lambda_3(t)$ de cara robô são atualizadas iterativamente. Dessa forma, o comportamento repulsivo esperado é mantido.



(a) 1.0s (b) 2.5s (c) 4.0s

(d) 5.0s (e) 6.5s (f) 8.0s

Figura 5.8: Sequência de quadros ilustrando o comportamento repulsivo gerado pelas trilhas de feromônio no decorrer do tempo de execução do PheroSLAM.

Além dos ajustes de direção calculados com o auxílio do algoritmo LPCS, deve-se estabelecer um mecanismo para evitar colisões entre os robôs e limites impostos no ambiente explorado. Como a abordagem do PheroSLAM utiliza

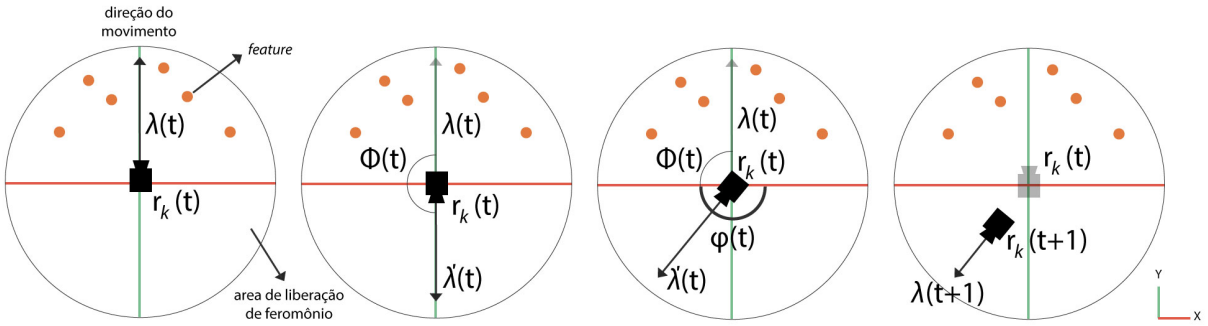


Figura 5.9: Mecanismo implementado no PheroSLAM para o desvio de obstáculos utilizando o campo de visão.

apenas o campo de visão pura como sensor para estimativa de posicionamento dos robôs, o referido mecanismo para desvio de obstáculos também deve considerar o campo de visão para que os obstáculos dispostos no ambiente sejam evitados.

Para resolver o problema supracitado, um simples mecanismo de desvio foi implementado. É importante salientar que o objetivo desta dissertação não é aprimorar técnicas para o desvio de obstáculos em ambientes desconhecidos. A Figura 5.9 ilustra o mecanismo desenvolvido com detalhes.

O mecanismo é baseado na detecção de um conjunto de *features* S_f que estão próximas de um robô $r_k(t)$, considerando o limite de alcance do sensor de depósito de feromônio deste robô. Tem-se como princípio evitar o conjunto de *features* S_f através de um ajuste de direção $\Phi(t)$.

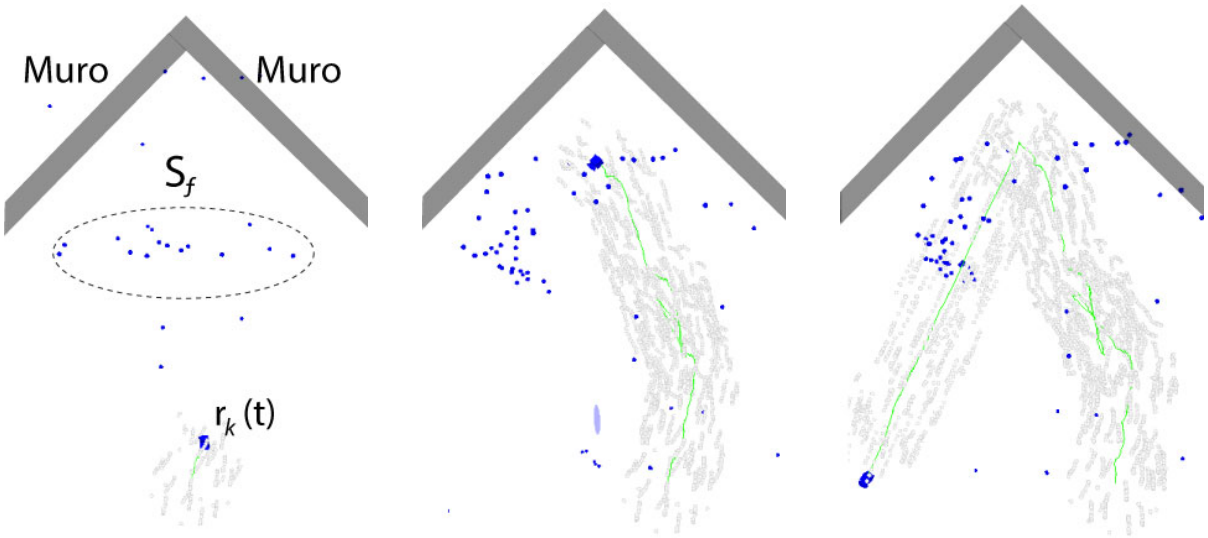
Observa-se, inicialmente que o robô $r_k(t)$ identifica um conjunto de *features* S_f (esferas na cor laranja) à sua frente. O conjunto S_f representa um possível obstáculo, dada a concentração de *features*. Assim, tem-se como intuito atualizar a direção de movimento $\lambda_k(t)$ do robô $r_k(t)$, de forma que a região na qual o conjunto S_f foi identificado seja evitada.

Para tal, um ajuste de direção $\lambda(t)$ deve ser calculado. Com isso, quando o robô $r_k(t)$ identifica o conjunto S_f o ajuste $\Phi(t)$ corresponde inicialmente a uma rotação de 180^o. O vetor de velocidade angular ω^R do robô $r_k(t)$ é atualizado para que a rotação supracitada seja realizada na próxima iteração.

Entretanto, para evitar que o robô $r_k(t)$ retorne pelo caminho contrário à direção de movimento $\lambda_k(t)$, desviando completamente do objetivo T_k, um novo ajuste de direção aleatório $\Phi_k(t)$ é calculado para que uma nova direção de movimento $\lambda'_k(t)$ seja obtida. O ajuste aleatório mencionado consiste em uma rotação máxima que considera um ângulo de rotação $\varphi(t)$.

Com isso, o vetor de velocidade angular ω^R do robô $r_k(t)$ é novamente atualizado. Na próxima iteração, por meio da atualização do vetor ω^R, o robô $r_k(t+1)$ possui uma nova direção de movimento $\lambda(t+1)$. A direção de movimento $\lambda(t+1)$ tende a evitar que o robô $r_k(t)$ retorne para uma área visitada recentemente.

A Figura 5.10 mostra uma situação na qual o mecanismo para desvio de obstáculos discutido é aplicado. Trata-se de um experimento realizado em um ambiente simulado na ferramenta Gazebo. Pode-se observar que, ao chegar próximo a um conjunto de *features* S_f, o robô desvia deste conjunto e uma nova direção é tomada. Os quadros dispostos na figura foram capturados em instantes de tempo específicos (segundos). Os pontos azuis na figura indicam a posição estimada das *features* identificadas. A trilha de feromônios gerada pelo PheroSLAM também pode ser visualizadas através dos pontos cinzas.



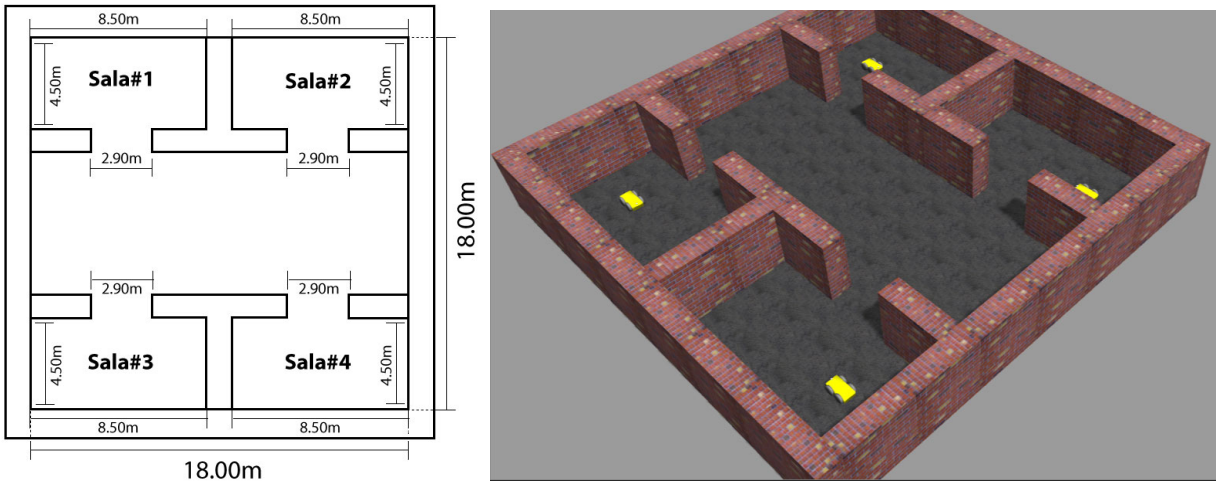
(a) Quadro 1 (1.0s) (b) Quadro 2 (6.5s) (c) Quadro 3 (13.0s)

Figura 5.10: Sequência de quadros que ilustram o comportamento do mecanismo de desvio de obstáculos implementado no PheroSLAM.

Considerando o que foi proposto no presente capítulo para a abordagem PheroSLAM, alguns experimentos práticos em um ambiente simulado foram realizados a fim validar a proposta. Os resultados experimentais obtidos são detalhados na seção seguinte.

5.5 Resultados experimentais

A fim de validar a robustez da proposta do PheroSLAM, alguns experimentos foram definidos considerando um ambiente interno simulado mais complexo. O ambiente construído pode ser analisado na Figura 5.11. Observa-se que o ambiente discutido possui 4 salas com dimensões idênticas e um corredor central de acesso a estas salas. Durante a construção deste ambiente na ferramenta Gazebo, teve-se como objetivo definir um ambiente no qual pode-se visualizar o comportamento repulsivo dos robôs com maior clareza.



(a) (b)

Figura 5.11: Ambiente interno simulado na ferramenta Gazebo para a execução dos algoritmos MonoSLAM e PheroSLAM.

Considerando o ambiente simulado de experimentação descrito na Figura 5.11, os seguintes experimentos (Figura 5.12) foram realizados utilizando os algoritmos MonoSLAM e PheroSLAM:

- **Experimentos 4, 5 e 6**: Os experimentos 4, 5 e 6 consistem em percorrer rotas pré-definidas (não-autônomas) utilizando 1, 2 e 4 robôs, respectivamente. Nestes experimentos, cada robô executa uma instância distinta do algoritmo MonoSLAM para gerar apenas dados de odometria. Os trajetos ilustrados na Figura 5.12 são pré-definido na implementação deste experimento. Nesse sentido, os robôs não tem como responsabilidade definir uma rota autônoma de navegação. Nos experimentos em questão, deseja-se observar as estimativas de posicionamento preditas, a fim de

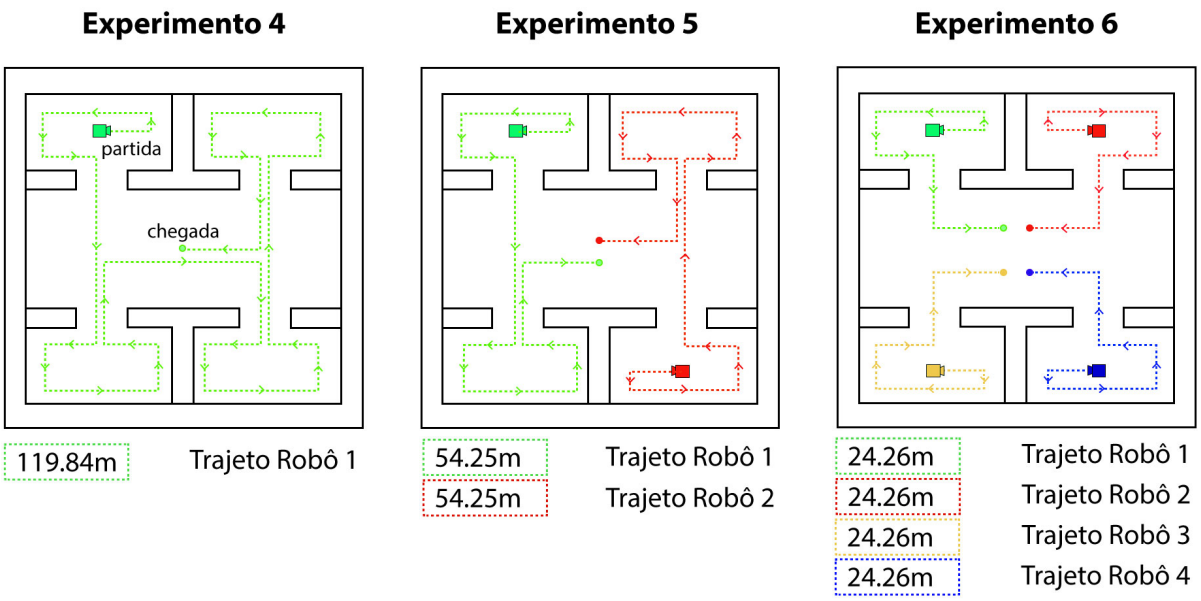


Figura 5.12: Trajetos planejados (não-autônomos) para execução dos experimentos 4, 5 e 6 utilizando apenas o algoritmo MonoSLAM.

verificar a robustez do algoritmo MonoSLAM em um ambiente de experimentação mais complexo;

- **Experimentos 7, 8 e 9**: Os experimentos 7, 8 e 9 consistem em posicionar um número específico de robôs em um ambiente desconhecido para que este seja explorado de forma autônoma, conforme ilustrado na Figura 5.13. Para tal, cada um dos robôs deve executar uma instância distinta do PheroSLAM. Nestes experimentos, adotou-se um período máximo de 15 minutos de exploração. Deseja-se observar as estimativa de posicionamento preditas pelos robôs e verificar a robustez da proposta do sistema desenvolvido.

Após a finalização dos experimentos 4, 5 e 6, observou-se que o algoritmo MonoSLAM gerou estimativas de posicionamento próximas à realidade, condizendo com os trajetos planejados ilustrados na Figura 5.12. Nesse sentido, considera-se que o algoritmo proposto para a predição de odometria visual é adequado para os ambientes de experimentação definidos.

Os resultados obtidos em cada um dos experimentos podem ser observados na Figura 5.14. Os dados coletados durante a execução dos experimentos em questão estão organizados na Tabela 5.1.

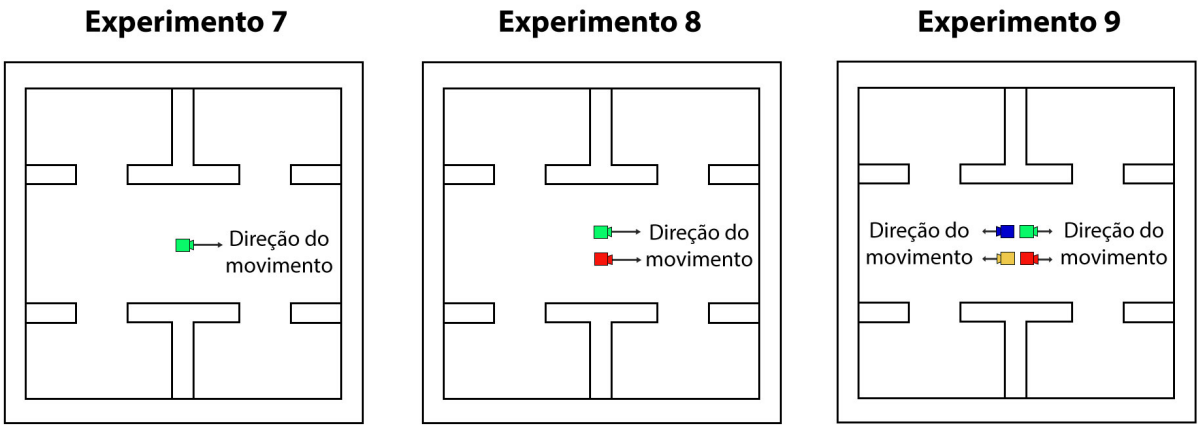


Figura 5.13: Ambiente de execução dos experimentos 7, 8 e 9.

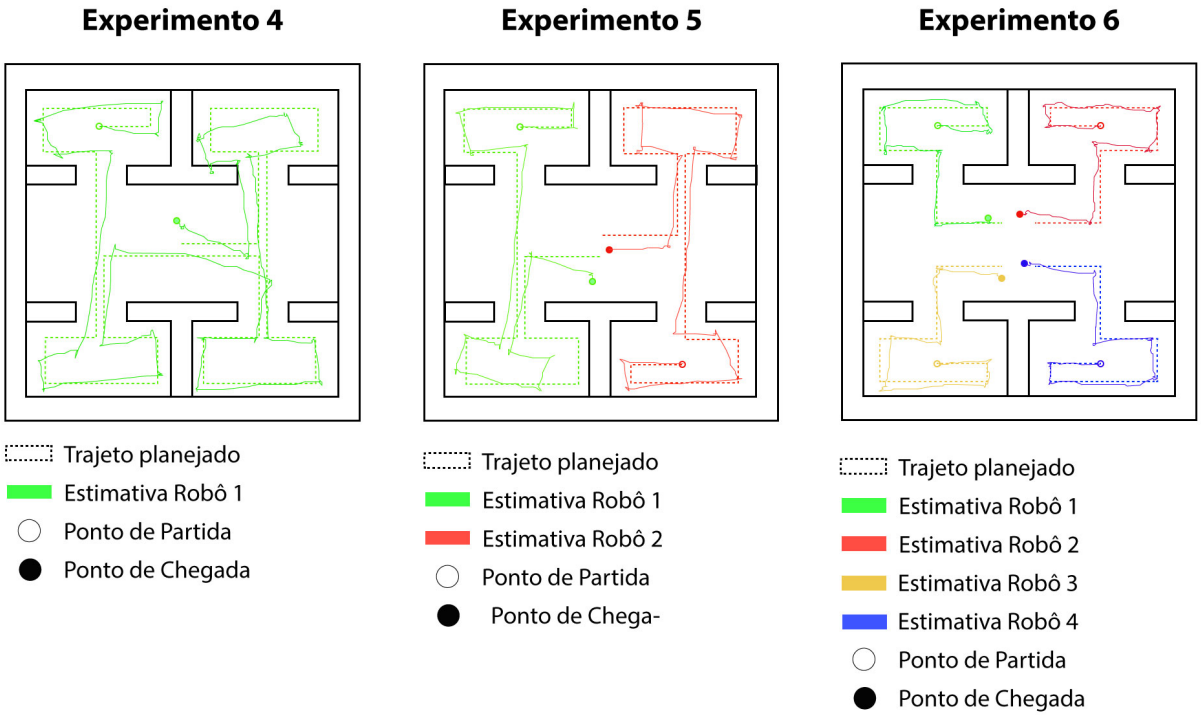


Figura 5.14: Resultados obtidos com os Experimento 4, 5 e 6, nos quais um ou mais robôs exploram o ambiente a partir de uma trajetória pré-definida (não-autônoma) executando apenas o algoritmo MonoSLAM.

Experimentos		Quant. de *features* capturadas	Quant. de *features* removidas	Dist. percorrida/ Dist. trajeto (metros)	Tempo de execução (m:s)
Exp. 4	Robô 1	193	97	139.25/119.84	13:14
Exp. 5	Robô 1	147	59	60.03/54.25	6:24
	Robô 2	151	63	60.83/54.25	6:27
Exp.6	Robô 1	118	26	29.31/24.26	2:45
	Robô 2	123	32	30.53/24.26	2:47
	Robô 3	120	29	28.59/24.26	2:43
	Robô 4	119	28	28.25/24.26	2:42

Tabela 5.1: Dados obtidos nos experimentos 4, 5 e 6, utilizando o algoritmo MonoSLAM.

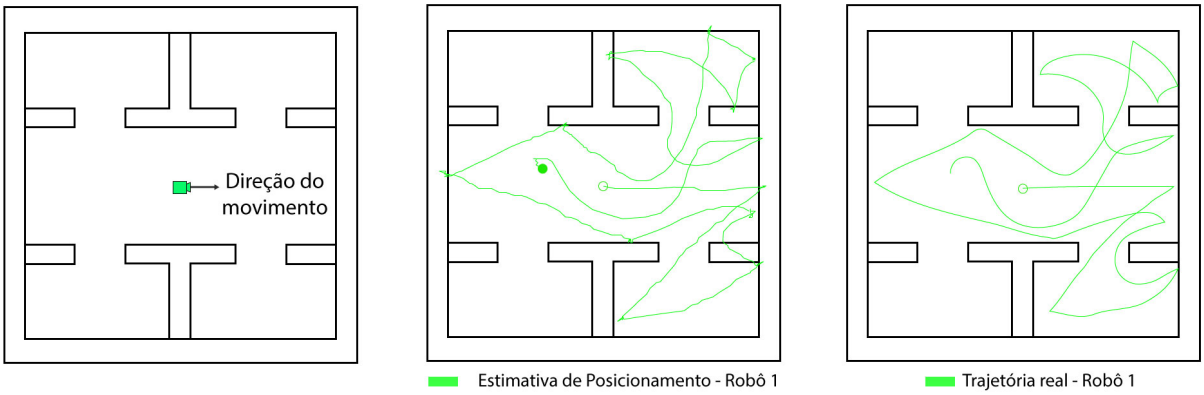


Figura 5.15: Resultados obtidos com o Experimento 7, no qual 1 robô explora o ambiente de forma autônoma utilizando o PheroSLAM.

Os experimentos 7, 8 e 9 descritos anteriormente foram executados considerando a tarefa de exploração. Vale ressaltar que tal tarefas foi executada de forma autônoma, uma vez que cada um dos robôs simulados executou uma instância do PheroSLAM. Os resultados referentes aos experimentos 7, 8 e 9 podem ser observados nas figuras 5.15, 5.16 e 5.17, respectivamente.

As Tabelas 5.2, 5.3 e 5.4 organizam as distâncias percorridas e estimadas por cada robô. Além das distâncias, podem ser analisados os quantitativos de *features* capturadas e removidas do sistema e a distância que cada robô percorreu em cada um dos cômodos do ambiente.

Pode-se observar nos experimentos realizados que os erros de estimativa de posicionamento acumulados reduzem à medida que uma quantidade maior de robôs é empregada à uma tarefa. Ainda assim, os erros gerados não compro-

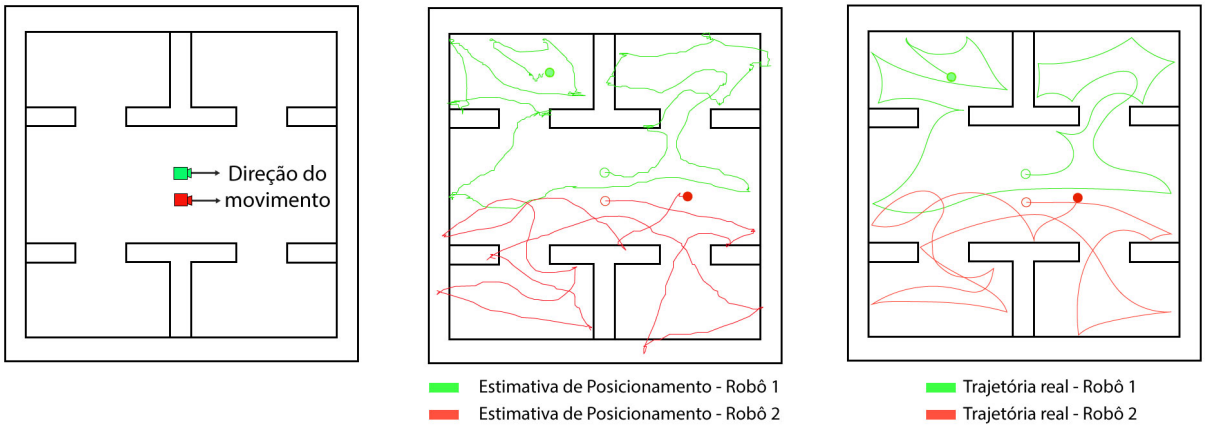


Figura 5.16: Resultados obtidos com o Experimento 8, no qual 2 robôs exploram o ambiente de forma autônoma utilizando o PheroSLAM.

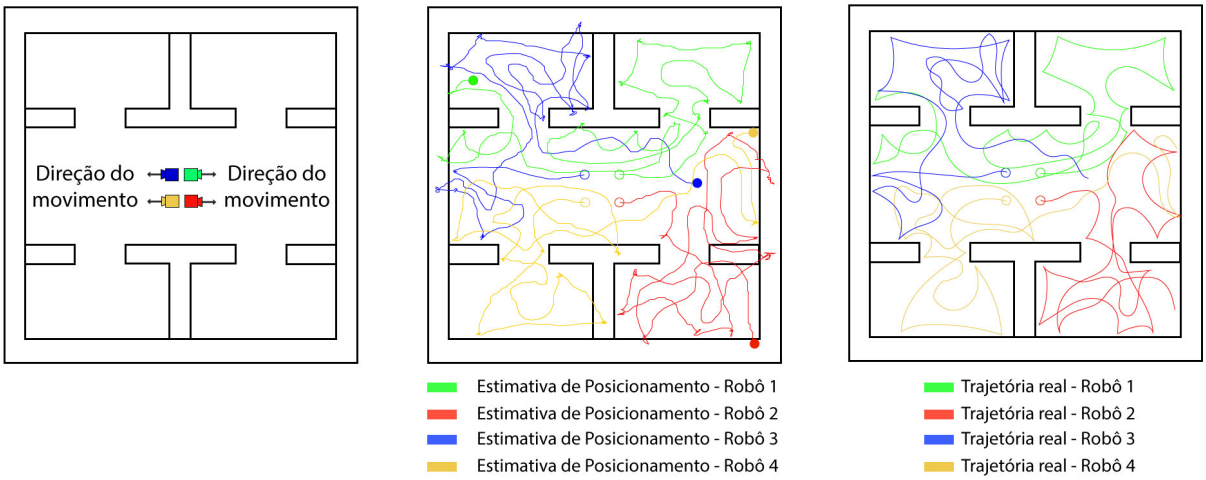


Figura 5.17: Resultados obtidos com o Experimento 9, no qual 4 robôs exploram o ambiente de forma autônoma utilizando o PheroSLAM.

Experimento 7								
Robôs	**Sala 1**	**Sala 2**	**Sala 3**	**Sala 4**	**Corredor**	**Dist. total percorrida / Dist. estimada**	***Features* capturadas**	***Features* removidas**
Robô 1	0.0m	24.26m	0.0m	19.75m	75.68m	119.69m / 138.83m	204	112

Tabela 5.2: Dados obtidos no Experimento 7.

Experimento 8								
Robôs	**Sala 1**	**Sala 2**	**Sala 3**	**Sala 4**	**Corredor**	**Dist. total percorrida / Dist. estimada**	***Features* capturadas**	***Features* removidas**
Robô 1	30.19m	26.77m	0.0m	0.0m	40.99m	97.95m / 128.46m	156	60
Robô 2	0.0m	0.0m	29.12m	19.30m	44.80m	93.22m / 111.91m	159	61

Tabela 5.3: Dados obtidos no Experimento 8.

meteram a realização das tarefas de exploração.

Pode-se concluir, também, que quanto maior o número de robôs empregados, maior é a cobertura do ambiente explorado. No Experimento 7 (Figura 5.15), nota-se que o ambiente não foi completamente explorado durante o período de exploração proposto. Dessa forma, assume-se que um tempo maior deve ser considerado para que o ambiente seja completamente explorado.

Entretanto, no Experimento 9 (Figura 5.17), todas as salas do ambiente são visitadas no tempo proposto. Observa-se, ainda, uma tendência de vigilância, uma vez que o comportamento repulsivo gerado pelas trilhas de feromônio fazem com que os robôs distanciem-se uns dos outros.

Com isso, a distribuição espacial dos robôs aproxima-se de uma distribuição ideal, na qual cada robô responsabiliza-se pela cobertura e vigilância de áreas específicas. Tal comportamento evita, ainda, a aglomeração de robôs em uma mesma área.

Nesse sentido, os resultados obtidos mostraram que o mecanismo bio-inspirado desenvolvido alcançou os objetivos esperados, uma vez que foi capaz de criar o comportamento repulsivo discutido neste capítulo. Dessa forma, conclui-se que o PheroSLAM é capaz de guiar robôs em tarefas de exploração e vigilância de ambientes internos desconhecidos.

5.6 Considerações finais

O presente capítulo apresentou uma abordagem bio-inspirada, denominada PheroSLAM, baseada em uma versão modificada da abordagem Colônia

Experimento 9								
Robôs	**Sala 1**	**Sala 2**	**Sala 3**	**Sala 4**	**Corredor**	**Dist. total percorrida / Dist. estimada**	***Features* capturadas**	***Features* removidas**
Robô 1	5.58m	19.14m	0.0m	0.0m	56.54m	81.26m / 103.70m	119	21
Robô 2	0.0m	0.0m	0.0m	32.62m	52.89m	85.51m / 109.59m	123	27
Robô 3	47.31m	0.0m	0.0m	0.0m	37.24m	85.55m / 103.29m	124	27
Robô 4	0.0m	0.0m	50.35m	0.0m	28.97m	79.32m / 105.22m	120	24

Tabela 5.4: Dados obtidos no Experimento 9.

de Formigas, capaz de guiar robôs autônomos em tarefas de exploração e vigilância de ambientes internos. Além disso, apresentou-se uma arquitetura de integração entre o mecanismo bio-inspirado e uma extensão do algoritmo MonoSLAM, capaz de prover dados de odometria visual para múltiplos robôs utilizando apenas visão monocular.

Experimentos foram realizados em um ambiente interno simulado visando verificar a robustez da proposta de implementação. Os resultados obtidos mostraram que a abordagem desenvolvida é capaz de guiar múltiplos robôs em tarefas de exploração e vigilância de ambientes internos.

CAPÍTULO

6

Conclusão

A presente dissertação apresentou um sistema bio-inspirado, baseado em uma versão modificada da abordagem Colônia de Formigas, para a exploração e vigilância de ambientes utilizando múltiplos robôs. Também propôs-se uma extensão do algoritmo MonoSLAM que é capaz de prover dados de odometria visual para múltiplos robôs que exploram ambientes internos de forma autônoma.

Experimentos foram realizados em ambientes reais e simulados a fim de validar a robustez da abordagem proposta, na qual duas frentes de trabalho foram identificadas, sendo estas:

1. A extensão do algoritmo MonoSLAM enquanto mecanismo para obtenção de dados de odometria visual considerando ambientes e tarefas com múltiplos robôs;

2. O desenvolvimento de um sistema bio-inspirado, baseado na abordagem IAS-SS (Calvo et al., 2011), para coordenar múltiplos robôs em tarefas de exploração e vigilância de ambientes desconhecidos.

Os resultados experimentais obtidos mostraram, inicialmente, que o algoritmo MonoSLAM é suficientemente robusto para prover dados de odometria visual em um ambiente controlado interno para múltiplos robôs.

Uma outra contribuição importante identificada é a possibilidade de manter mapas probabilísticos 3D utilizando o MonoSLAM. Com isso, aumenta-se a possibilidade de aplicações em outros ambientes, uma vez que Veículos Aéreos Não-Tripulados podem ser empregados em tarefas mais complexas.

O sistema bio-inspirado PheroSLAM proposto também foi alvo de experimentação extensiva, a fim de validar a proposta de coordenação descrita no Capítulo 5. Os resultados experimentais obtidos a partir da utilização do PheroSLAM mostraram que o sistema proposto é suficientemente robusto para coordenar pequenas equipes de robôs em ambientes internos controlados. Evidências empíricas evidenciaram que o PheroSLAM apresenta boa dispersibilidade, o que promove o aumento da cobertura de ambientes em tarefas de exploração e vigilância.

A junção das abordagens MonoSLAM e IAS-SS se mostrou satisfatória durante a realização dos experimentos propostos nesta dissertação. Entretanto, o algoritmo MonoSLAM apresentou limitações relativas ao quantitativo de *features* que podem ser mantidas em um mapa. Outras limitações foram identificadas e poderão ser alvo de estudo em trabalhos futuros, como a criação de um mecanismo mais elaborado para desvio de obstáculos.

Os resultados obtidos durante a etapa de experimentação foram submetidos como artigos científicos para os seguintes eventos:

- 13^{th} Mexican International Conference on Artificial Intelligence - MICAI 2014;
- 30^{th} ACM/SIGAPP Symposium On Applied Computing - SAC 2015;
- International Conference on Autonomous Agents Multiagent Systems - AAMAS 2015.

Os artigos científicos submetidos para os eventos MICAI 2014 e SAC 2015 não foram aceitos para publicação, contudo, foram bem avaliados em algumas das revisões realizadas. Vale ressaltar que tais contribuições foram essenciais para o alinhamento e continuidade do trabalho. O artigo submetido ao evento AAMAS 2015 ainda está sendo avaliado para eventual publicação.

6.1 Trabalhos futuros

Dadas as contribuições e limitações apresentadas, pretende-se estender, em trabalhos futuros, a proposta de sistema aqui apresentada para ambientes esparsos abertos e dinâmicos, considerando múltiplos VANTs. Novos experimentos poderão ser realizados a fim de verificar o comportamento da abordagem proposta em ambientes abertos e dinâmicos, considerando que o algoritmo MonoSLAM utilizando para prover dados de odometria visual possui limitações com relação à quantidade de *features* que podem ser mantidas em um mapa. Considera-se, assim, que outras abordagens de SLAM visual podem ser avaliadas.

Acredita-se que novas pesquisas devem ser realizadas a fim de expandir a base científica e o arcabouço tecnológico existente na área de robótica colaborativa bio-inspirada. Tais avanços são essenciais para que o campo de visão pura seja aplicado, de forma robusta, em aplicações colaborativas com múltiplos robôs em ambientes dinâmicos e abertos.

Referências Bibliográficas

Aulinas, J., Petillot, Y., Salvi, J., e Lladó, X. (2008). The slam problem: A survey. In *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, páginas 363–371, Amsterdam, The Netherlands, The Netherlands. IOS Press. Citado 2 vezes nas páginas 2 e 55.

Ballesta, M., Gil, A., Reinoso, O., Juliá, M., e Jiménez, L. M. (2010). Multi-robot map alignment in visual slam. *WTOS*, 9(2):213–222. Citado na página 22.

Betke, M. e Gurvits, L. (1995). Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13:135–142. Citado na página 14.

Brekke, E. e Chitre, M. (2013). Bayesian multi-hypothesis scan matching. In *OCEANS - Bergen, 2013 MTS/IEEE*, páginas 1–10. Citado na página 3.

Burgard, W., Fox, D., e Thrun, S. (1997). Active mobile robot localization. In *In Proceedings of IJCAI-97. IJCAI, Inc.* Morgan Kaufmann. Citado na página 11.

Burgard, W., Moors, M., Fox, D., Simmons, R., e Thrun, S. (2000). Collaborative multi-robot exploration. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*. Citado na página 19.

Buschka, P. (2006). *An Investigation on Hybrid Maps for Mobile Robots*. PhD thesis, Institutionen for Teknik, Orebro University. Citado na página 16.

Calvo, R. (2012). *Sistemas bio-inspirados para coordenação de múltiplos robôs móveis*. PhD thesis, Universidade de São Paulo. Citado 4 vezes nas páginas xiv, 47, 53, e 54.

Calvo, R., de Oliveira, J. R., Figueiredo, M., e Romero, R. A. F. (2011). Inverse aco applied for exploration and surveillance in unknown environments. In *The Third International Conference on Advanced Cognitive Technologies and Applications*, páginas 142–147, Rome, Italy. Citado 8 vezes nas páginas 5, 48, 49, 52, 54, 58, 65, e 81.

Campbell, J., Sukthankar, R., Nourbakhsh, I., e Pahwa, A. (2005). A robust visual odometry and precipice detection system using consumer-grade monocular vision. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, páginas 3421–3427. Citado na página 5.

Carlone, L., Du, J., Kaouk Ng, M., Bona, B., e Indri, M. (2014). Active slam and exploration with particle filters using kullback-leibler divergence. *Journal of Intelligent & Robotic Systems*, 75(2):291–311. Citado na página 3.

Chaudhury, S. (2011). Disaster management using mobile robots. In *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief*, ACWR '11, páginas 108–108, New York, NY, USA. ACM. Citado na página 1.

Cho, H. e Kim, S.-W. (2010). Mobile robot localization using biased chirp-spread-spectrum ranging. *Industrial Electronics, IEEE Transactions on*, 57(8):2826–2835. Citado 2 vezes nas páginas 3 e 10.

Choset, H. e Nagatani, K. (2001). Topological simultaneous localization and mapping (slam): Toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17:125–137. Citado na página 15.

Clipp, B., Lim, J., Frahm, J.-M., e Pollefeys, M. (2010). Parallel, real-time visual slam. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, páginas 3961–3968. Citado na página 26.

Davison, A. J. (2003). Real-time simultaneous localization and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE Inter-*

national Conference on, volume 2, páginas 1403–1410. Citado 2 vezes nas páginas 17 e 18.

Davison, A. J., Reid, I. D., Molton, N. D., e Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067. Citado 15 vezes nas páginas xiii, 5, 6, 11, 12, 15, 16, 17, 18, 21, 26, 27, 28, 29, e 56.

de Castro, L. (2006). *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*. Chapman & Hall/CRC Computer & Information Science Series. Taylor & Francis. Citado 2 vezes nas páginas 47 e 48.

de la Puente, P. e Rodriguez-Losada, D. (2014). Feature based graph-slam in structured environments. *Autonomous Robots*, 37(3):243–260. Citado na página 10.

den Besten, M., Stützle, T., e Dorigo, M. (2000). Ant colony optimization for the total weighted tardiness problem. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J., e Schwefel, H.-P., editors, *Parallel Problem Solving from Nature PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, páginas 611–620. Springer Berlin Heidelberg. Citado na página 49.

Deneubourg, J.-L., Aron, S., Goss, S., e Pasteels, J. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168. Citado na página 51.

Di Caro, G. e Dorigo, M. (1998). Antnet: Distributed stigmergetic control for communications networks. *J. Artif. Int. Res.*, 9(1):317–365. Citado na página 49.

Dorigo, M. (1992). *Optimization, learning and natural algorithms*. PhD thesis, Dipartimento di Elettronica, Polit. di Milano. Citado 8 vezes nas páginas 5, 47, 48, 49, 52, 57, 58, e 64.

Dorigo, M. e Stützle, T. (2003). The ant colony optimization metaheuristic: Algorithms, applications, and advances. In Glover, F. e Kochenberger, G., editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research Management Science*, páginas 250–285. Springer US. Citado 3 vezes nas páginas 48, 49, e 50.

Elfes, A. (1990). Sonar-based real-world mapping and navigation. In Cox, I. e Wilfong, G., editors, *Autonomous Robot Vehicles*, páginas 233–249. Springer New York. Citado na página 13.

Elinas, P., Sim, R., e Little, J. (2006). /spl sigma/slam: stereo vision slam using the rao-blackwellised particle filter and a novel mixture proposal distribution. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, páginas 1564–1570. Citado na página 17.

Forster, C., Lynen, S., Kneip, L., e Scaramuzza, D. (2013). Collaborative monocular slam with multiple micro aerial vehicles. In *IROS*, páginas 3962–3970. IEEE. Citado na página 22.

Franklin, S. e Graesser, A. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, ECAI '96, páginas 21–35, London, UK, UK. Springer-Verlag. Citado 2 vezes nas páginas 2 e 3.

Gambardella, L. M. e Dorigo, M. (2000). An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS J. on Computing*, 12(3):237–255. Citado na página 49.

Gambardella, L. M., Taillard, E., e Agazzi, G. (1999). Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. RelatÃş-rio tÃľcnico. Citado na página 49.

Gutmann, J. e Konolige, K. (1999). Incremental mapping of large cyclic environments. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, páginas 318–325, Monterey, California. Citado na página 11.

Han, J.-S., Ji, S.-H., Kim, K.-H., Lee, S.-M., e Choi, B.-W. (2011). Collective robot behavior controller for a security system using open sw platform for a robotic services. In *Control, Automation and Systems (ICCAS), 2011 11th International Conference on*, páginas 1402–1404. Citado 2 vezes nas páginas 22 e 52.

Jensfelt, P. (2001). *Approaches to Mobile Robot Localization in Indoor Environments*. PhD thesis, Royal Institute of Technology. Citado na página 14.

Jiménez Lugo, J. e Zell, A. (2014). Framework for autonomous on-board navigation with the ar.drone. *J. Intell. Robotics Syst.*, 73(1-4):401–412. Citado na página 58.

Johannsson, H., Kaess, M., Fallon, M., e Leonard, J. (2013). Temporally scalable visual slam using a reduced pose graph. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, páginas 54–61. Citado na página 26.

Johnson, R. e Pilcher, M. G. (1988). The traveling salesman problem, edited by e.l. lawler, j.k. lenstra, a.h.g. rinnooy kan, and d.b shmoys, john wiley sons, chichester, 1985, 463 pp. *Networks*, 18(3):253–254. Citado na página 49.

Karlsson, N., Di Bernardo, E., Ostrowski, J., Goncalves, L., Pirjanian, P., e Munich, M. (2005). The vslam algorithm for robust localization and mapping. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, páginas 24–29. Citado 2 vezes nas páginas 4 e 26.

Kootstra, G. e Schomaker, L. R. B. (2009). Using symmetrical regions of interest to improve visual slam. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'09, páginas 930–935, Piscataway, NJ, USA. IEEE Press. Citado na página 26.

Kundu, A., Krishna, K. M., e Jawahar, C. V. (2010). Realtime motion segmentation based multibody visual slam. In *Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing*, ICVGIP '10, páginas 251–258, New York, NY, USA. ACM. Citado na página 26.

Lategahn, H., Geiger, A., e Kitt, B. (2011). Visual slam for autonomous ground vehicles. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, páginas 1732–1737. Citado na página 26.

Lee, D., Kim, D., Lee, S., Myung, H., e Choi, H.-T. (2013). Experiments on localization of an auv using graph-based slam. In *URAI*, páginas 526–527. IEEE. Citado na página 3.

Lee, S. e Lee, S. (2013). Embedded visual slam: Applications for low-cost consumer robots. *Robotics Automation Magazine, IEEE*, 20(4):83–95. Citado na página 26.

Lee, Y.-J. e Song, J.-B. (2008). Visual slam in indoor environments using autonomous detection and registration of objects. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, páginas 671–676. Citado na página 26.

Maniezzo, D., Dorigo, M., Maniezzo, V., e Colorni, A. (1991). Ant system: An autocatalytic optimizing process. RelatÃşrio tÃl'cnico, Dipartimento di Elettronica, Politecnico di Milano, Italy. Citado na página 49.

Markham, L., Melton, S. C., e Dodds, Z. (2007). Robot control via region-based 3d reconstruction. In *Proceedings of the 10th IASTED International Conference on Intelligent Systems and Control*, ISC '07, páginas 29–34, Anaheim, CA, USA. ACTA Press. Citado na página 1.

Mellinger, D., Michael, N., e Kumar, V. (2014). Trajectory generation and control for precise aggressive maneuvers with quadrotors. In Khatib, O., Kumar, V., e Sukhatme, G., editors, *Experimental Robotics*, volume 79 of *Springer Tracts in Advanced Robotics*, páginas 361–373. Springer Berlin Heidelberg. Citado na página 58.

Montemerlo, M., Thrun, S., Koller, D., e Wegbreit, B. (2002a). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada. AAAI. Citado 2 vezes nas páginas 3 e 10.

Montemerlo, M., Thrun, S., Koller, D., e Wegbreit, B. (2002b). Fastslam: a factored solution to the simultaneous localization and mapping problem. In *Eighteenth national conference on Artificial intelligence*, páginas 593–598, Menlo Park, CA, USA. American Association for Artificial Intelligence. Citado 2 vezes nas páginas 20 e 25.

Montemerlo, M., Thrun, S., Koller, D., e Wegbreit, B. (2003). FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico. IJCAI. Citado 2 vezes nas páginas 3 e 10.

Mozos, O. M., Gil, A., Ballesta, M., e Reinoso, O. (2007). Current topics in artificial intelligence. chapter Interest Point Detectors for Visual SLAM, páginas 170–179. Springer-Verlag, Berlin, Heidelberg. Citado na página 26.

Murphy, K. (2010). Bayesian map learning in dynamic environments. In *In Neural Info. Proc. Systems (NIPS)*, páginas 1015–1021. MIT Press. Citado 2 vezes nas páginas 3 e 10.

Myung, H., Lee, H.-K., Choi, K., e Bang, S. (2010). Mobile robot localization with gyroscope and constrained kalman filter. *International Journal of Control, Automation and Systems*, 8(3):667–676. Citado 2 vezes nas páginas 3 e 10.

Negenborn, R. (2003). *Robotic Localization and Kalman Filters*. PhD thesis, Utrecht University. Citado na página 12.

Neira, J., Ribeiro, M. I., e Tardos, J. D. (1997). Mobile robot localization and map building using monocular vision. *Proc. Int'l Symp. Intelligent Robotics Systems*. Citado na página 21.

Newcombe, R. A., Lovegrove, S., e Davison, A. J. (2011). DTAM: dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, páginas 2320–2327. Citado na página 3.

Newman, P. e Ho, K. (2005). Slam-loop closing with visually salient features. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, páginas 635–642. Citado 2 vezes nas páginas 17 e 25.

Newman, P., Leonard, J., Tardos, J., e Neira, J. (2002). Explore and return: experimental validation of real-time concurrent mapping and localization. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, páginas 1802–1809. Citado 2 vezes nas páginas 20 e 25.

Olson, C. F. (2002). Selecting landmarks for localization in natural terrain. *Auton. Robots*, 12(2):201–210. Citado na página 15.

Pierce, D. e Kuipers, B. (1994). Learning to explore and build maps. In *Menlo Park*, páginas 1264–1271. AAAI/MIT Press. Citado na página 15.

Pinagé, F., Carvalho, J. R. H., Freitas, E. R. V., e Neto, J. P. d. Q. (2013). Feature transform technique for combining landmark detection and tracking of

visual information of large rain forest areas. In *Proceedings of the 2013 IEEE Latin American Robotics Symposium*, LARS '13, páginas 30–37, Washington, DC, USA. IEEE Computer Society. Citado na página 58.

Reinelt, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag, Berlin, Heidelberg. Citado na página 49.

Royer, E., Lhuillier, M., Dhome, M., e Lavest, J.-M. (2007). Monocular vision for mobile robot localization and autonomous navigation. *Int. J. Comput. Vision*, 74(3):237–260. Citado 5 vezes nas páginas 2, 4, 16, 17, e 18.

Russo, L., Rosa, S., Bona, B., e Matteucci, M. (2014). A ros implementation of the mono-slam algorithm. *International Journal of Computer Science & Information Technology*, 6(1). Citado 2 vezes nas páginas 30 e 31.

Schmidt, D. e Berns, K. (2013). Climbing robots for maintenance and inspections of vertical structures-a survey of design aspects and technologies. *Robot. Auton. Syst.*, 61(12):1288–1305. Citado na página 1.

Sheng, W., Yang, Q., Tan, J., e Xi, N. (2006). Distributed multi-robot coordination in area exploration. *Robot. Auton. Syst.*, 54(12):945–955. Citado na página 57.

Shubina, K. e Tsotsos, J. K. (2010). Visual search for an object in a 3d environment using a mobile robot. *Comput. Vis. Image Underst.*, 114(5):535–547. Citado na página 1.

Sim, R., Elinas, P., e Griffin, M. (2005). Vision-based slam using the rao-blackwellised particle filter. In *In IJCAI Workshop on Reasoning with Uncertainty in Robotics*. Citado na página 3.

Simmons, R., Apfelbaum, D., Burgard, W., Fox, D. an Moors, M., Thrun, S., e Younes, H. (2000). Coordination for multi-robot exploration and mapping. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Austin, TX. AAAI. Citado 3 vezes nas páginas 18, 19, e 25.

Smith, R., Self, M., e Cheeseman, P. (1990). Autonomous robot vehicles. In Cox, I. J. e Wilfong, G. T., editors, *Autonomous Robot Vehicles*, chapter Estimating Uncertain Spatial Relationships in Robotics, páginas 167–193. Springer-Verlag New York, Inc., New York, NY, USA. Citado 2 vezes nas páginas 3 e 10.

Smith, R. C. e Cheeseman, P. (1986a). On the representation and estimation of spatial uncertainly. *Int. J. Rob. Res.*, 5(4):56–68. Citado 2 vezes nas páginas 11 e 13.

Smith, R. C. e Cheeseman, P. (1986b). On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4). Citado na página 10.

Steder, B., Grisetti, G., Stachniss, C., e Burgard, W. (2008). Visual slam for flying vehicles. *Robotics, IEEE Transactions on*, 24(5):1088–1093. Citado na página 3.

Stützle, T. e Dorigo, M. (1999). New ideas in optimization. chapter ACO Algorithms for the Quadratic Assignment Problem, páginas 33–50. McGraw-Hill Ltd., UK, Maidenhead, UK, England. Citado na página 49.

Thrun, S. (2002). Particle filters in robotics. In *in Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI*. Citado na página 14.

Thrun, S., Burgard, W., e Fox, D. (2000). A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, páginas 321–328. Citado 2 vezes nas páginas 13 e 19.

Thrun, S., Burgard, W., e Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press. Citado 2 vezes nas páginas 11 e 12.

Thrun, S. e Montemerlo, M. (2005). The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430. Citado na página 10.

Turduev, M., Cabrita, G., KÄśrtay, M., Gazi, V., e Marques, L. (2012). Experimental studies on chemical concentration map building by a multi-robot system using bio-inspired algorithms. *Autonomous Agents and Multi-Agent Systems*, páginas 1–29. Citado na página 1.

U. Lima, P. e M. Custódio, L. (2005). Multi-robot systems. In Patnaik, S., C. Jain, L., G. Tzafestas, S., Resconi, G., e Konar, A., editors, *Innovations in Robot Mobility and Control*, volume 8 of *Studies in Computational Intelligence*, páginas 1–64. Springer Berlin Heidelberg. Citado na página 55.

Wang, T.-C. e Chen, C.-H. (2013). Improved simultaneous localization and mapping by stereo camera and surf. In *Automatic Control Conference (CACS), 2013 CACS International*, páginas 204–209. Citado na página 3.

Williams, B. e Reid, I. (2010). On combining visual slam and visual odometry. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, páginas 3494–3500. Citado na página 26.

Wulf, O., Arras, K., Christensen, H., e Wagner, B. (2004). 2d mapping of cluttered indoor environments by means of 3d perception. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, páginas 4204–4209 Vol.4. Citado 2 vezes nas páginas 17 e 25.

Yamauchi, B., Schultz, A., e Adams, W. (1998). Mobile robot exploration and map-building with continuous localization. In *In Proceedings of the 1998 IEEE/RSJ International Conference on Robotics and Automation*, páginas 3715–3720. Citado na página 10.

Yin, J., Carlone, L., Rosa, S., e Bona, B. (2014). Graph-based robust localization and mapping for autonomous mobile robotic navigation. In *Mechatronics and Automation (ICMA), 2014 IEEE International Conference on*, páginas 1680–1685. Citado na página 10.

Zhang, B., Liu, J., e Chen, H. (2013). Amcl based map fusion for multi-robot slam with heterogenous sensors. In *Information and Automation (ICIA), 2013 IEEE International Conference on*, páginas 822–827. Citado na página 3.

Zou, Danping e Tan, P. (2013). Coslam: Collaborative visual slam in dynamic environments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(2):354–366. Citado 2 vezes nas páginas 4 e 21.