Arthur Vicente Ribacki

Comparação experimental de métodos de exploração de ambientes desconhecidos usando robôs móveis autônomos

Porto Alegre

2011

Arthur Vicente Ribacki

Comparação experimental de métodos de exploração de ambientes desconhecidos usando robôs móveis autônomos

Trabalho de Diplomação

Orientador:
Prof. Dr. Edson Prestes e Silva Jr.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

Porto Alegre

2011

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

RIA	Robot Institute of America,	p. 11
AGV	Automatic Guided Vehicle,	p. 11
PVC	Problemas de Valor de Contorno,	p. 12
SRT	Sensor-based Random Tree,	p. 12
RRT	Rapidly Exploring Random Tree,	p. 16
SOR	Sucessivas Sobre-Relaxações,	p. 24
HIMM	Histogramic In-Motion Mapping,	p. 26
RSL	Região Segura Local,	p. 29
ARIA	Advanced Robot Interface for Applications,	p. 33
IfTEA	Interface for Testing Exploation Algorithms,	p. 36

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho se situa no ramo da robótica móvel autônoma, mais especificamente na área de exploração de ambientes desconhecidos. Para solucionar o problema de exploração, o robô deve ser capaz de decidir quais direções ele deve seguir de modo a obter um modelo completo do ambiente utilizando o retorno de seus diferentes sensores.

São apresentados dois métodos de exploração com abordagens bem distintas. O primeiro utiliza soluções de Problemas de Valor de Contorno (PVC) para guiar o robô até as fronteiras do ambiente conhecido e o segundo é baseado no método SRT (*Sensor-based Random Tree*), que utiliza a geração aleatória e incremental de uma estrutura de dados em forma de árvore. É realizada uma comparação por simulação e experimentos em um robô real, a fim de ressaltar as vantagens e desvantagens de cada algoritmo em uma aplicação real.

As simulações são realizadas em ambientes densos e esparsos, utilizando-se diferentes valores para velocidade máxima. Com estes resultados, é feita uma análise quantitativa em termos de caminho percorrido e tempo de execução. Os experimentos utilizam o robô Pioneer P3-DX da Adept Mobile Robots, equipado com um sensor do tipo laser, e permitiram realizar uma análise qualitativa dos algoritmos.

Palavras-Chave: robótica móvel autônoma, exploração de ambientes desconhecidos, Sensor-based Random Tree, Problemas de Valor de Contorno.

ABSTRACT

Experimental comparison of methods for exploring unknown environments using autonomous mobile robots

This work is in the field of autonomous mobile robotics, more specifically in the area of exploration of unknown environments. To solve the problem of exploration, the robot must be able to decide which direction it should follow in order to obtain a complete model of the environment using the return of its different sensors.

We present two exploration methods with very different approaches. The first uses solutions of Boundary Value Problems (BVP) to guide the robot to the borders of the known environment and the second method is based on the SRT (Sensor-based Random Tree), which uses the generation of a random, incremental tree structure. A comparison is performed by simulation and experiments in a real robot in order to highlight the advantages and disadvantages of each algorithm in a real application.

The simulations are performed in dense and sparse environments, using different values for maximum speed. With these results, a quantitative analysis is made in terms of path length and runtime. The experiments uses the robot Pioneer P3-DX from Adept Mobile Robots, equipped with a laser rangefinder, and allowed a qualitative analysis of the algorithms.

Keyworkds: autonomous mobile robots, exploration of unknown ervironments, Sensor-based Random Tree, Boundary Value Problems.

1 INTRODUÇÃO

Antes mesmo de a palavra robô ter sido cunhada, o homem sonhava com a construção de um servo autônomo, incansável e obediente. Na antiguidade, lendas e mitos descrevem criaturas animadas feitas de metal e argila capazes de feitos extraordinários. Diversos autômatos foram construídos e projetados ao longo da história e os avanços tecnológicos do século XX e XXI nos deixam cada vez mais próximos da literatura fantástica de Isaac Asimov. Mesmo que ainda não existam robôs humanoides completamente autônomos circulando entre nós, os robôs já desempenham um papel fundamental em nossa sociedade, que abriga mais de um bilhão de robôs operacionais (WORLD ROBOTICS, 2011).

Uma definição oficial de robô é dada pela RIA (*Robot Institute of America*) (SPONG et al., 2006, p. 2):

> Um robô é um manipulador multifuncional e reprogramável projetado para mover materiais, partes, ferramentas ou dispositivos especializados através de movimentos variáveis programados para o desempenho de uma variedade de tarefas.

Este conceito está relacionado aos robôs industriais, que tiveram seu advento na década de 60 na forma de braços robóticos, como o robô Unimate instalado em 1961 em uma fábrica da General Motors, e veículos guiados automaticamente (AGV, em inglês) , utilizados para o transporte de carga. Naquela época, o conceito de robô era menos ligado ao conceito de inteligência artificial e mais ao conceito de dispositivo reprogramável, como expresso pela definição da RIA. Seguindo os avanços da microeletrônica e informática, os robôs hoje possuem maior poder de processamento e autonomia. Atualmente as tarefas que envolvem o uso de robôs vão desde limpezas domésticas até exploração espacial, objetivando minimizar a exposição do ser humano em ambientes perigosos ou substituí-lo em tarefas repetitivas, tediosas ou que exijam alta precisão. Exemplos dessas tarefas incluem a exploração de Marte (NASA, 2003), medidas de níveis de radiação, como no desastre ocorrido na usina de Fukushima (GIRALDI, 2011), e assistência a operações cirúrgicas, efetuadas por intermédio do robô DaVinci da Intuitive Surgical (2001).

Um robô autônomo deve ser capaz de interagir com o ambiente, percebendo-o através de seus sensores e modificando-o com seus atuadores. As ações tomadas são resultado do processamento dos sensores, dando ao robô uma noção de raciocínio, associando-lhe uma inteligência artificial. Esta autonomia tem um papel fundamental na consolidação de robôs destinados a uso profissional e pessoal. Exemplos de robôs profissionais incluem os veículos aéreos não tripulados para uso militar e robôs de campo como os robôs de ordenha da DeLaval (1998). Exemplos de robôs pessoais incluem robôs domésticos como aspiradores da iRobot (2002) e cortadores de grama da Robomow (1998); robôs de entretenimento e lazer, incluindo robôs brinquedos como os da WowWee (2004) e sistemas para educação e pesquisa como o robô NAO da Aldebaran Robotics (2005).

Estes tipos de robôs trazem uma mudança de paradigma na robótica. Como os exemplos demonstram, eles não estão mais restritos ao chão de fábrica, já que interagem com o ambiente e seres humanos. Este trabalho foca na exploração de ambientes desconhecidos. Para atuar e atingir seus objetivos, o robô deve ter conhecimento de seu ambiente de trabalho. Este pode ser informado diretamente ao robô, pré-programado, ou descoberto de maneira iterativa pelo mesmo, caracterizando um processo de exploração.

Serão estudados dois algoritmos de exploração. O primeiro utiliza soluções de Problemas de Valor de Contorno (PVC) para guiar o robô até as fronteiras (PRESTES, 2003). O segundo é baseado no método SRT *(Sensor-based Random Tree)* , que utiliza a geração aleatória e incremental de uma estrutura de dados em forma de árvore (ORIOLO et al., 2004). Estes foram escolhidos por possuírem abordagens bem distintas. Será realizada uma comparação por simulação e experimentos em um robô real, a fim de ressaltar suas vantagens e desvantagens em uma aplicação real.

O trabalho é dividido da seguinte maneira. O capítulo 2 apresenta uma visão geral da área da robótica móvel. O capítulo 3 apresenta os conceitos de exploração de ambientes e os algoritmos estudados. O capítulo 4 apresenta os experimentos realizados e os resultados obtidos. Finalmente, o capítulo 5 apresenta as conclusões e os trabalhos futuros.

2 VISÃO GERAL

A robótica é uma área multidisciplinar e uma visão geral precisaria abordar conceitos como cinemática, dinâmica, mecânica, eletrônica, inteligência artificial entre muitos outros. Mantendo o foco na parte computacional da robótica móvel autônoma, este capítulo faz uma rápida introdução a conceitos importantes que serão necessários para compreensão do que segue.

Para que um robô seja plenamente autônomo, é importante que sua atuação independa do ambiente em que ele está inserido. Para isso, ele deve ser capaz de determinar sua posição no mundo (localização), ser capaz de construir uma representação interna do ambiente já conhecido (mapeamento) e ser capaz de determinar caminhos que o levem a posições desejadas (movimentação).

Os algoritmos atuais integram estas tarefas de diferentes maneiras (MAKARENKO et al., 2002), como sugere a Figura 2.1. Nas seções que seguem, são introduzidos os conceitos de mapeamento, localização, movimentação, SLAM e localização ativa; conceitos de exploração clássica e integrada são detalhados no capítulo 3.

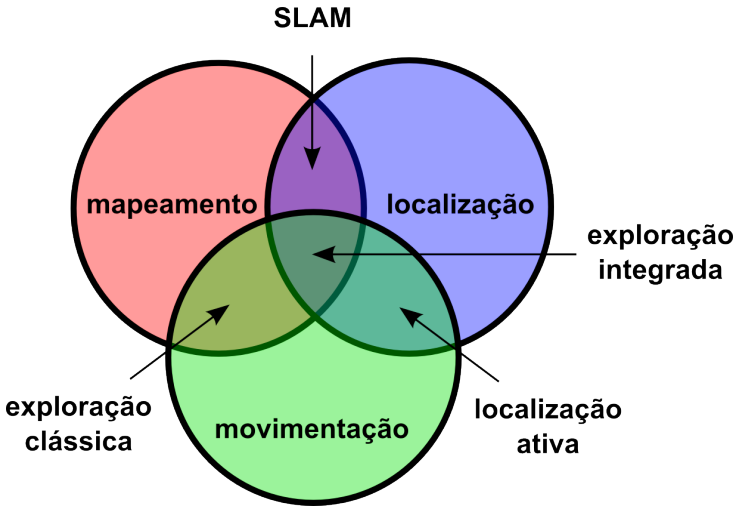
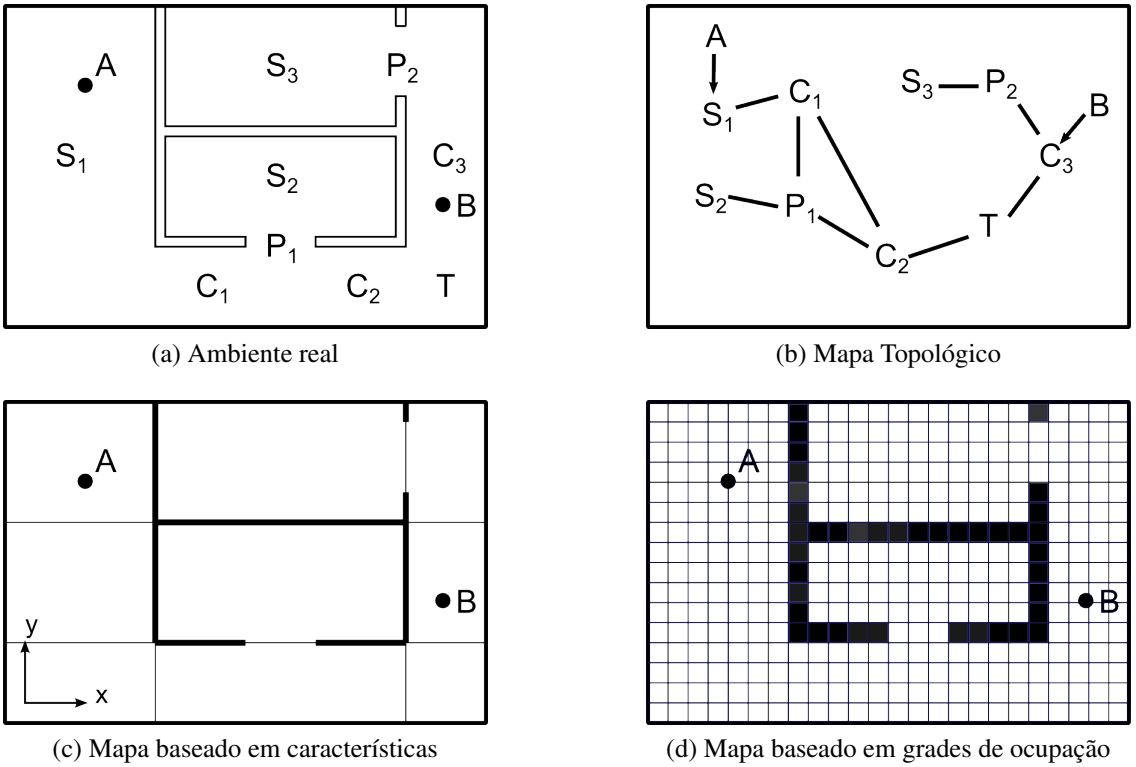


Figura 2.1: Tarefas envolvidas no processo de exploração.

Figura extraída de (MAKARENKO et al., 2002).

2.1 Mapeamento

A tarefa de mapeamento consiste em guardar em uma representação interna um modelo do mundo utilizando como entrada o retorno dos sensores. Podemos definir duas categorias de modelos: métricos e topológicos.



(a) Ambiente real

(b) Mapa Topológico

(c) Mapa baseado em características

(d) Mapa baseado em grades de ocupação

Figura 2.2: Exemplos de diferentes tipos de mapas.

Figura baseada em (FILLIAT; MEYER, 2003).

Os *mapas topológicos* descrevem o ambiente de forma mais abstrata, identificando entidades, suas características e correlações. A Figura 2.2b mostra um exemplo de mapa topológico, representado por um grafo onde os nodos são corredores, portas ou salas e as arestas contêm informações que permitem o deslocamento entre os diferentes locais. Como vantagens, podemos citar: (1) sua complexidade depende da complexidade do ambiente, (2) permite um planejamento eficiente e (3) não necessita de posições precisas do robô. No entanto, estes mapas são difíceis de construir e manter, além de não serem adequados a tarefas que exijam uma posição precisa.

Em *mapas métricos*, o ambiente é representado por um conjunto de objetos com coordenadas cartesianas. Seguindo esta ideia, podemos descrever o ambiente através de características extraídas a partir das informações fornecidas pelos sensores do robô, a fim de representar a forma e posição dos obstáculos. Essas características podem ser pontos específicos, bem como

linhas que representem o contorno dos obstáculos. Assim, o mundo da Figura 2.2a pode ser representado por retas como visto na Figura 2.2c. Uma dificuldade é extrair as características desejadas de leituras ruidosas, nem sempre confiáveis, e transformá-las em modelos métricos que possam ser utilizados.

Outra possibilidade é representar o espaço acessível ao robô usando malhas. A técnica mais difundida é a **grade de ocupação** (ELFES, 1989), vista na Figura 2.2d, onde cada célula representa uma porção retangular do ambiente visto na Figura 2.2a. Cada célula guarda uma probabilidade de estar ocupada, a qual é utilizada para classificar a região como: livre, ocupada ou desconhecida. A grande vantagem deste tipo de mapa é a facilidade de manipulação e a rápida atualização utilizando diferentes sensores, enquanto que os principais problemas são granularidade, escalabilidade e extensibilidade, consequências do tamanho fixo da célula (BAILEY; NEBOT, 2001). A complexidade deste tipo de mapa não depende da complexidade do ambiente, apenas do número de total de células, o qual é determinado pelo tamanho do mapa e tamanho da célula. Quanto menor o tamanho da célula, maior o nível de detalhes do ambiente que o mapa será capaz de representar, em troca de um maior custo em memória e poder computacional para mapear uma área de mesma dimensão. Existem diversas técnicas para atualização e representação da probabilidade de ocupação de cada célula como certeza fuzzy, bayes, heurística, gaussiana ou frequência.

As vantagens e desvantagens de mapas métricos e topológicos são ortogonais. Para o processo de exploração, a facilidade de construção e manutenção desempenha um papel importante; sendo assim, uma parte considerável dos algoritmos de exploração opta por mapas métricos, mais especificamente pelas grades de ocupação. Outra abordagem é utilizar ambos os tipos de mapas, tirando proveito de suas vantagens complementares e reduzindo suas limitações (THRUN, 1998).

2.2 Movimentação

A movimentação consiste na determinação de caminhos e trajetórias entre uma postura inicial q_{inicio} e uma postura final q_{fim}. Para um ambiente planar ($\mathbb{R}^2$), a postura de um robô móvel pode ser definida pela tupla (x, y, θ), i.e., sua posição e orientação. Considerando $\mathcal{Q}$ o conjunto de todas as posturas possíveis e $\mathcal{Q}_{livre}$ o conjunto de todas as posturas nas quais o robô não colide com obstáculos, a solução para um problema de *planejamento de caminhos* é uma função contínua $c(s)$ parametrizada entre zero e um, tal que:

$$c(s) : [0,1] \rightarrow \mathcal{Q}, \text{ onde } c(0) = q_{inicio},\ c(1) = q_{fim} \text{ e } c(s) \in \mathcal{Q}_{livre} \forall s \in [0,1] \tag{2.1}$$

Isto quer dizer que a solução é uma sequência de posturas que o robô deve assumir de modo que ele saia de q_{inicio} e chegue a q_{fim} sem colidir com obstáculos. Esta função não informa como realizar o percurso, mas apenas a forma que o percurso deve possuir.

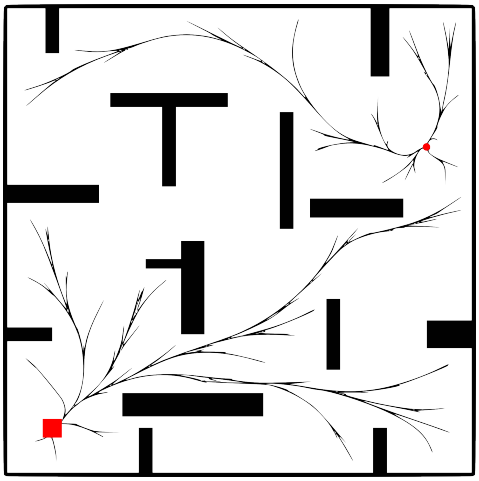
Se o caminho for parametrizado em função do tempo t, então $c(t)$ é uma *trajetória*, e a velocidade e aceleração podem ser calculadas pela primeira e segunda derivada em relação ao tempo. Assim, além de descrever a forma, uma trajetória descreve *como* o percurso deve ser realizado.

Distinguem-se dois grandes grupos de técnicas de planejamento: busca em grafos e campos de potenciais (SIEGWART et al., 2011, p. 373). Algoritmos de planejamento baseados em busca em grafos são normalmente divididos em duas etapas: modelamento do espaço livre e acessível por um grafo conexo e determinação de um caminho neste grafo que permita o robô atingir a posição objetivo. A construção dos grafos pode ser feita de diferentes maneiras: grafo de visibilidade (LOZANO-PÉREZ; WESLEY, 1979), diagrama de Voronoi (AURENHAMMER, 1991), decomposição em células exatas (CHOSET et al., 2000), decomposição em células aproximadas (SIEGWART et al., 2011, p. 287–290). Finalmente, um algoritmo de busca é utilizado para identificar a sucessão de nodos que resulta em um caminho, preferencialmente ótimo, entre o nodo inicial e final. Exemplos desses algoritmos são: busca em largura, busca em profundidade, Dijkstra, A* (HART et al., 1968) e D* (STENTZ, 1994).

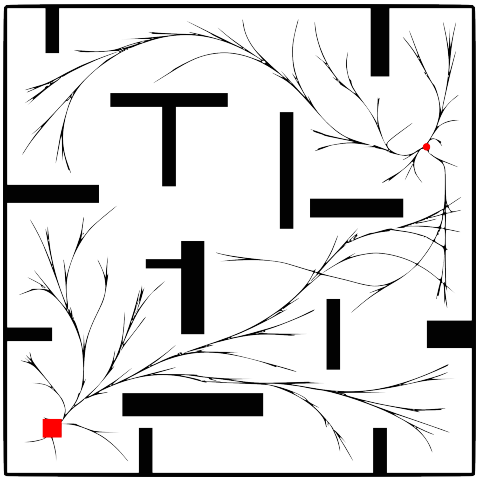
Outra abordagem consiste em fazer a busca em um grafo randômico, como exemplo as RRTs (*Rapidly Exploring Random Tree*) (LAVALLE; KUFFNER, 2001). Não é requerida a decomposição prévia do ambiente, apenas um mapa contendo informações dos obstáculos, uma vez que o grafo é construído durante o processo de planejamento. A cada passo, uma configuração aleatória q_{rand} pertencente ao espaço livre $\mathcal{Q}_{livre}$ é escolhida. Em seguida, determina-se o nodo pertencente à árvore mais próximo de q_{rand}, chamado de $q_{proximo}$. Partindo de $q_{proximo}$, um ramo de tamanho fixo é expandido em direção de q_{rand}. A posição final, q_{nova}, é adicionada na árvore se o ramo não colidir com obstáculos. Eventualmente, será gerado um nodo suficientemente próximo da posição destino, terminando o algoritmo. Apesar de não produzir uma solução ótima para o problema, este algoritmo é muito rápido computacionalmente, sendo utilizado para problemas complexos que requerem um planejamento em várias dimensões, como em braços manipuladores (SIEGWART et al., 2011, p. 386).

Uma otimização consiste em expandir duas RRT: uma a partir da origem e outra a partir

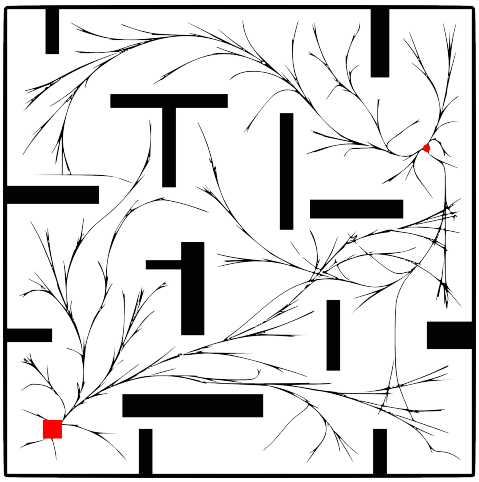
do destino, acelerando o processo. Um exemplo de expansão utilizando a RRT bidirecional (LAVALLE; KUFFNER, 1999) é visto na Figura 2.3. Estados intermediários após 500 e 1000 iterações são mostrados nas Figuras 2.3a e 2.3b, respectivamente. Após 1582 iterações (Figura 2.3c) o caminho final é computado (Figura 2.3d).



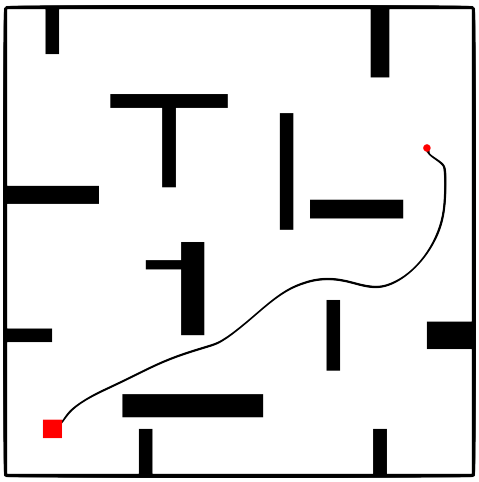
(a) Após 500 iterações



(b) Após 1000 iterações



(c) Finalizado após 1582 iterações



(d) Caminho resultante

Figura 2.3: Evolução de um planejamento RRT bidirecional. A posição inicial é representada pelo quadrado e a posição final pelo disco.

Figura extraída de (LAVALLE; KUFFNER, 1999).

Outro conjunto de técnicas tem como base o uso de funções de potenciais. Para melhor compreensão, faz-se analogia a potenciais elétricos. Nesta analogia, o robô é visto como uma partícula móvel carregada positivamente, em um ambiente onde obstáculos são cargas positivas fixas, exercendo forças de repulsão, e a posição objetivo é uma carga negativa fixa, a qual exerce uma força de atração. Calculando o gradiente do potencial definido pelo ambiente, obtém-se um campo vetorial que define o caminho até a posição objetivo a partir de qualquer posição do ambiente.

2.3 Localização

Para que um robô atinja seus objetivos, é fundamental saber sua localização no ambiente, pois para que ele saiba como chegar a uma determinada posição ele deve primeiro saber onde se encontra. Para isso, ele deve utilizar as informações oriundas de seus sensores proprioceptivos ou extraceptivos. Sensores *proprioceptivos* fornecem informações internas ao sistema, e.g. temperatura, velocidade, aceleração, número de rotações das rodas; enquanto *extraceptivos* fornecem informações do ambiente, e.g. distância dos obstáculos, intensidade luminosa, amplitude sonora.

A localização pode ser classificada como local ou global. A *localização local* consiste em estimar a postura (x, y, θ) do robô a partir de uma postura conhecida. Note que para o processo de construção de mapas métricos costuma-se associar a postura inicial ao ponto $(0, 0, 0)$. Desse modo, o robô parte de uma postura conhecida ao iniciar a exploração. O uso apenas de técnicas de odometria leva ao método conhecido como *dead-reckoning*, que consiste em integrar o número de rotações das rodas a fim de estimar o deslocamento relativo do robô e consequentemente sua postura atual. No entanto, por envolver uma integração, esta técnica está sujeita ao acúmulo de erros sistemáticos e não sistemáticos. Erros *sistemáticos*, como desalinhamento ou diferença de tamanho das rodas, são determinísticos e podem ser corrigidos por uma correta calibração do sistema. Erros *não sistemáticos*, como derrapagem, não podem ser previstos e adicionam incertezas na localização. Assim, a utilização exclusiva de dados proprioceptivos funciona apenas a curto prazo e a estimativa precisa ser corrigida regularmente através de informações do ambiente. A Figura 2.4a mostra um exemplo de um mapa gerado apenas com o uso de odometria, enquanto no mapa da Figura 2.4b é utilizada uma técnica de localização. Note como o mapa sofre deformações devido ao erro na estimação da localização do robô.

Na *localização global*, o robô não tem conhecimento prévio de sua postura. A partir das leituras de seus sensores e em posse de um mapa do ambiente, ele determina neste mapa a postura mais provável de produzir uma resposta sensorial igual à observada. A localização global sofre de um problema conhecido como *perception aliasing*, i.e., para um dado sistema sensorial, diferentes posições do mapa podem gerar respostas muito semelhantes, fazendo com que dois lugares distintos pareçam iguais, como ilustrado na Figura 2.5. Este problema é oriundo da quantidade limitada de informações que um sistema sensorial consegue prover e da quantidade limitada de informação disponível no ambiente. Por exemplo, ao ser colocado em um labirinto com paredes idênticas, a eficiência do sistema de localização humano decai rapidamente, pois este dependente fortemente da visão, que não é capaz de extrair informações suficientes. Se considerarmos as limitações dos sensores dos robôs, percebemos a amplitude do problema de

(a) Mapa construído apenas com técnicas de odometria

(b) Mapa construído utilizando técnicas de localização

Figura 2.4: Mapas construídos com (a) apenas odometria e (b) usando técnicas de localização.

Figura extraída de (THRUN, 1998).

localização: o que para os humanos é uma tarefa trivial, para os robôs é um grande labirinto de paredes brancas.

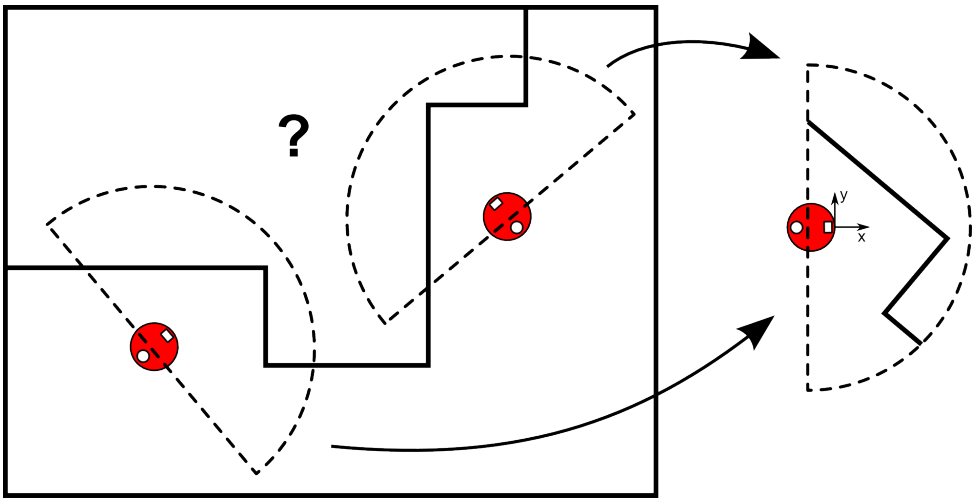


Figura 2.5: *Perception aliasing*: o robô possui o mesmo retorno sensorial em posturas distintas.

Para amenizar esse problema, ao invés de estimar uma única postura a cada iteração (localização baseada em um *única hipótese*), várias prováveis posturas são rastreadas, cada uma com uma probabilidade associada (localização baseada em *múltiplas hipóteses*). Conforme o robô avança e recebe mais informações, algumas posturas possuem suas probabilidades incrementadas e outras diminuídas, reduzindo a ambiguidade da localização. Quando o robô deve tomar uma decisão com base na sua localização, uma técnica simples consiste em utilizar uma média ponderada das possíveis posições com base nas probabilidades associadas.

2.4 Localização e Mapeamento Simultâneo

As técnicas de localização e mapeamento simultâneo, conhecidas por SLAM (*Simultaneous Localization and Mapping*), têm o seguinte objetivo: partindo de uma posição desconhecida, em um ambiente desconhecido, o robô deve ser capaz de construir um mapa, utilizando apenas observações relativas do ambiente, e utilizar este mapa incompleto para se localizar simultaneamente. A dificuldade do problema vem do fato de um mapa ser necessário para estimar a posição do robô ao mesmo tempo em que a posição é necessária para a construção do mapa.

Alguns algoritmos de SLAM precisam ser executados após a completa aquisição dos dados, devido à sua alta complexidade. Essa classe de algoritmos é dita *off-line*, enquanto aqueles capazes de executar durante a aquisição dos dados são chamados *online*. As técnicas mais difundidas para solucionar o problema de SLAM são baseadas no uso do Filtro de Kalman e do Filtro de Partículas.

2.5 Localização Ativa

A *localização ativa* assume que a rotina de localização tem acesso total ou parcial sobre os atuadores do robô, provendo a oportunidade de aumentar a eficiência e robustez da localização (BURGARD et al., 1997). O objetivo é determinar onde o robô deve ir e em que direção ele deve orientar seus sensores de forma a melhor localizar sua posição.

Um trabalho importante nesta área foi desenvolvido por Burgard et al. (1997) que definiram um método de localização ativa com base na localização de Markov, um método passivo. Na localização de Markov, a posição do robô é representada através de uma função densidade de probabilidade arbitrária, atualizada a partir dos dados sensoriais. Na localização ativa, as posições e orientações que o robô deve atingir são determinadas de modo a diminuir a entropia associada à função densidade de probabilidade.

3 EXPLORAÇÃO DE AMBIENTES DESCONHECIDOS

Integrando-se as tarefas de movimentação e mapeamento obtém-se um algoritmo de **exploração clássica**, definida como:

> Exploração é o ato de se mover através de um ambiente desconhecido construindo um mapa que pode ser usado para navegação subsequente. Uma boa estratégia de exploração é aquela que gera um mapa completo ou quase completo em um tempo razoável (YAMAUCHI, 1997).

Sendo assim, a ideia principal é minimizar o trajeto percorrido enquanto se maximiza a informação obtida. Note que este conceito não considera a *qualidade* do mapa. Sem considerarmos a certeza da localização, o mapa gerado pode ser inútil uma vez que erros de odometria vão sendo acumulados conforme o processo de exploração avança. Isto é exemplificado na Figura 2.4, onde o mapa 2.4a é obtido utilizando apenas técnicas de odometria e o mapa 2.4b utilizando uma técnica de SLAM.

Assim, além de maximizar a quantidade de informação obtida, uma boa estratégia de exploração deve considerar a *qualidade* da localização para determinar a trajetória a ser efetuada. Este tipo de estratégia é chamada de **exploração integrada** e tem como objetivo orientar o robô em um ambiente desconhecido, gerando um percurso que maximize a certeza de sua localização e permita um mapeamento completo e correto. Note que o comprimento do caminho não precisa ser o mínimo, pois pode ser necessário retornar a lugares já explorados para aumentar a certeza da localização.

Uma quantidade representativa de estratégias utiliza métodos de planejamento de caminhos para orientar o robô em direção a regiões desconhecidas do ambiente. Yamauchi (1997) introduziu o método conhecido como *exploração baseada em fronteiras*. Utilizando grades de ocupação, definiu-se uma fronteira como o limite entre a região explorada livre de obstáculos e a região não explorada, conforme visto na Figura 3.1. Com base na probabilidade de cada célula, estas são classificadas como livres, desconhecidas ou ocupadas 3.1a. Um processo aná-

logo à detecção de bordas e extração de regiões da computação gráfica é então utilizado para detectar as células entre o espaço livre e o espaço desconhecido (Figura 3.1b). Estas células são então agrupadas por adjacência formando regiões de fronteiras. Regiões com um tamanho mínimo (normalmente o tamanho do robô), são então consideradas fronteiras (Figura 3.1c). Após o processo de detecção, o robô navega até a fronteira **não visitada mais próxima**, atualiza a grade de ocupação e o processo se repete.

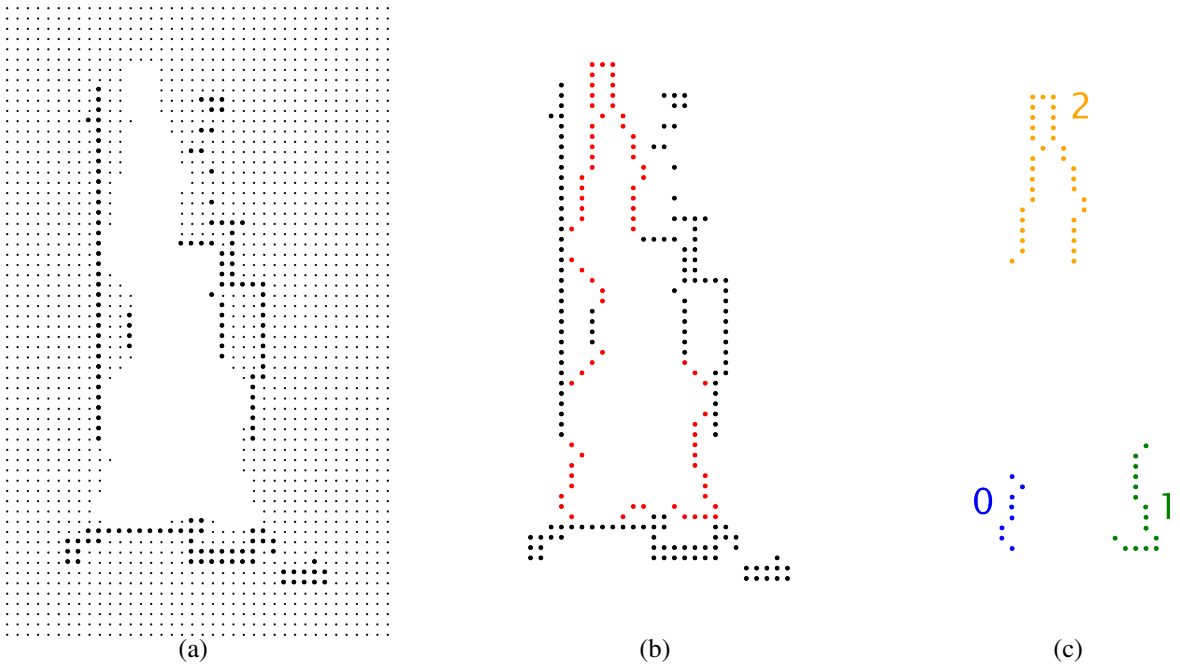


Figura 3.1: Detecção de fronteiras: (a) grades de ocupação, (b) células marcadas como fronteiras, (c) fronteiras agrupadas por adjacência com um tamanho maior que o mínimo.

Figura extraída de (YAMAUCHI, 1997).

Diversos métodos se baseiam no mesmo princípio, mas definem qual a fronteira que o robô deve atingir através de uma função de custo-benefício. O custo pode ser visto como a distância até a fronteira e o benefício pode levar em conta fatores como a expectativa de aumento do mapa, a observação de regiões específicas como portas ou corredores, ou até mesmo o aumento na certeza da localização. Note que estes critérios podem ser contraditórios, pois aumentar a certeza pode significar retroceder a lugares já visitados o que representa um ganho de informação nulo em termos de expansão do ambiente. Para contornar este dilema, muitos algoritmos de exploração integrada operam em dois estados: enquanto a certeza da localização estiver aceitável, ele orienta o robô em direção às regiões desconhecidas e faz o mapeamento utilizando um algoritmo de SLAM; quando a certeza da localização estiver abaixo de certo limiar, o robô é orientado a posições que diminuam a incerteza da localização, caracterizando um processo de localização ativa.

A seguir, descreveremos dois algoritmos de exploração clássica. O primeiro utiliza soluções de Problemas de Valor de Contorno (PVC) para guiar o robô até as fronteiras (PRESTES, 2003). O segundo é baseado no método SRT *(Sensor-based Random Tree)*, que utiliza a geração aleatória e incremental de uma estrutura de dados em forma de árvore (ORIOLO et al., 2004).

3.1 Exploração utilizando soluções de PVC

Um **problema de valor de contorno** é caracterizado por uma equação diferencial e um conjunto de restrições ditas condições de contorno. **Condições de contorno** especificam o valor que função e/ou suas derivadas devem assumir em *diferentes pontos*, contrariamente a **condições iniciais**, que especificam o valor que a função e/ou suas derivadas devem assumir em um *dado instante de tempo*. Uma solução para este problema deve ser uma solução da equação diferencial que satisfaça as condições fornecidas. As condições de contorno são normalmente especificadas para pontos pertencentes à fronteira do domínio. Condições de *Dirichlet* especificam o **valor na fronteira**, enquanto condições de *Neumann* especificam o **valor da derivada normal à fronteira**.

A equação diferencial utilizada pelo algoritmo é a equação de Laplace $\nabla^2 u = 0$, cujas soluções são chamadas de funções harmônicas. A equação também é conhecida como **equação do potencial**, sendo aplicada na modelagem de diferentes problemas como condução térmica, potenciais eletrostáticos e potenciais gravitacionais.

Primeiramente, introduz-se o algoritmo de planejamento de caminhos baseado em funções harmônicas que serviu como base para o algoritmo de exploração. Após isso, o algoritmo de exploração é apresentado e a seção termina apresentando algumas extensões do algoritmo original.

3.1.1 Planejamento de caminhos

O método de exploração aqui apresentado é uma extensão do método de planejamento de caminhos baseado em funções harmônicas de Connolly e Grupen (1993). Para o planejamento, é calculada a solução numérica da equação de Laplace $\nabla^2 p(\mathbf{r}) = 0$ com condições de contorno de Dirichlet, onde $p(\mathbf{r})$ é o potencial na posição $\mathbf{r} \in \mathbb{R}^2$. O ambiente é dividido em uma malha regular (grade de ocupação), onde cada célula representa uma porção quadrada do ambiente e armazena um valor de potencial. Células ocupadas (obstáculos) têm seu potencial definido em 1.0 (alto), enquanto células objetivos têm seu potencial definido em zero (baixo), caracterizando

as condições de contorno de Dirichlet. As células restantes têm seu potencial calculado através de iterações sucessivas utilizando-se um método de diferenças finitas. Uma característica importante é que o potencial gerado é livre de mínimos locais e os caminhos produzidos são suaves.

A Figura 3.2 apresenta o campo vetorial resultante após aplicação do planejamento de caminho descrito anteriormente. Células pretas representam obstáculos, células brancas regiões livres e a célula amarela a posição objetivo. Note como o robô, representado pelo polígno vermelho, é capaz de determinar um caminho para posição destino partindo de qualquer posição inicial apenas seguindo o gradiente do campo potencial.

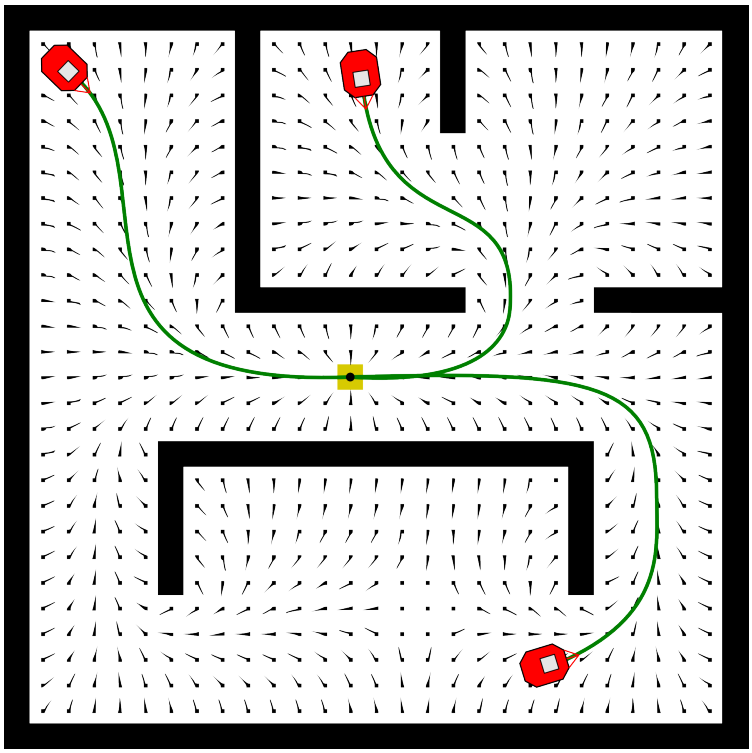


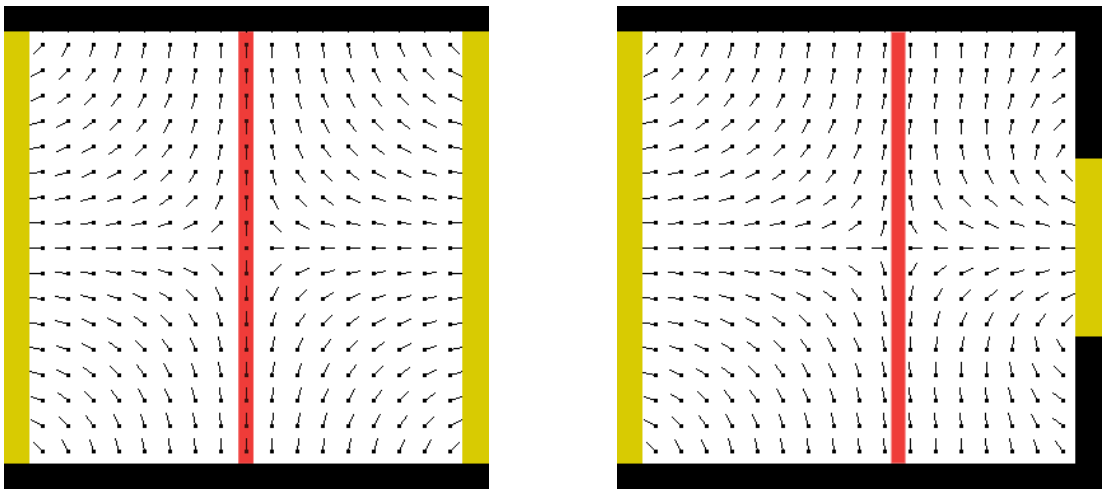
Figura 3.2: Campo vetorial gerado por um planejamento de caminhos baseado em PVC.

Para solucionar numericamente equações diferenciais, dois métodos são comumente utilizados (PRESTES, 2003 apud FORTUNA, 2000): o método Gauss-Seidel e o método de Sucessivas Sobre-Relaxações (SOR) . O primeiro é obtido pela discretização da equação diferencial por meio de diferenças centrais de segunda ordem. O segundo é uma expansão de Gauss-Seidel que introduz um termo extra para acelerar a convergência do método numérico.

3.1.2 Exploração de ambientes

O método de planejamento é estendido de forma a guiar o robô para as fronteiras do ambiente (PRESTES, 2003). Isto faz com que ao invés de definir uma única posição que o robô deve

atingir, sejam definidas várias posições que precisam ser visitadas, representadas pelas células amarelas na Figura 3.3a. A partir de qualquer ponto do ambiente está determinado um caminho que leva à fronteira mais vantajosa a partir daquele ponto. A natureza do potencial faz com que *vantajosa* nem sempre signifique a mais próxima ou a maior, mas uma relação entre tamanho e distância, conforme pode ser visto na Figura 3.3. Esta extensão leva ao algoritmo proposto por Prestes et al. descrito na Figura 3.4.



(a) Ambas com 17 células

(b) 17 contra 7 células

(c) 17 contra 3 células

Figura 3.3: Diferença de força de atração conforme tamanho e distância da fronteira. À esquerda da linha vermelha o robô visitará a fronteira esquerda e à direita da linha vermelha o robô visitará a fronteira direita.

A condição de parada *não explorou todo ambiente* (linha 2) consiste em verificar se não existe alguma **célula acessível e livre** ao lado de uma **célula não explorada**. Para verificar se uma célula livre é acessível ou não, foi utilizada uma variação do algoritmo de crescimento de regiões, que segmenta o espaço livre em uma região conectada (acessível) e outra desconectada (inacessível).

```
   Entrada: Robô a ser controlado
   Saída: Mapa do ambiente
1  início
2  |  enquanto não explorou todo ambiente faça
3  |  |  obtém a leitura dos sensores;
4  |  |  atualiza a probabilidade de ocupação das células dentro do campo de visão;
5  |  |  atualiza o potencial de cada célula;
6  |  |  calcula o vetor gradiente descendente a partir da sua posição corrente no mapa;
7  |  |  desloca-se seguindo a direção definida por este gradiente;
8  |  fim
9  fim
```

Figura 3.4: Algoritmo para exploração de ambientes utilizando PVC

As ações *obtém leitura dos sensores* e *atualiza probabilidade* (linhas 3 e 4) estão relacionadas com o método de mapeamento escolhido. O algoritmo original faz uso de uma técnica conhecida como HIMM (*Histogramic In-Motion Mapping*) (BORENSTEIN; KOREN, 1991) . Neste algoritmo, para cada retorno de um sensor, define-se uma linha de atualização, entre o ponto medido e postura corrente do robô. As células sob essa linha têm seu atributo de certeza reduzido em 1 (até um mínimo de zero), enquanto a célula no ponto medido têm seu atributo de certeza incrementado de 3 (até um máximo de 15). Quando este atributo for maior que dois, a célula é considerada **ocupada**.

A ação *atualiza potencial* (linha 5) é como no planejamento de caminhos: os obstáculos são definidos com potencial alto igual a um, enquanto que regiões desconhecidas (o objetivo) com potencial baixo igual à zero. Para o cálculo do potencial das demais células, utiliza-se o método Gauss-Seidel para resolução numérica da equação de Laplace, o que resulta na equação (3.1), onde $p^t_{i,j}$ é o potencial da célula de posição (i, j) na iteração t. Para o processo de exploração não é necessário que o potencial tenha convergido para que o robô comece a se deslocar e um número fixo de iterações é utilizado a cada atualização. É por utilizar este número fixo de iterações que o método SOR não é adequado: o termo adicionado para acelerar a convergência produz estados intermediários não confiáveis, podendo causar a colisão do robô.

$$p^{t+1}_{i,j} = \frac{1}{4}\left(p^{t+1}_{i-1,j} + p^{t}_{i+1,j} + p^{t+1}_{i,j-1} + p^{t}_{i,j+1}\right) \tag{3.1}$$

O vetor gradiente **v** (linha 6) é calculado utilizando:

$$\mathbf{v} = (v_x, v_y) = \left(\frac{p_{i-1,j} - p_{i+1,j}}{2}, \frac{p_{i,j-1} - p_{i,j+1}}{2}\right) \tag{3.2}$$

Finalmente, determina-se a direção ϕ necessária à ação *desloca-se* (linha 7) com:

$$\phi = arctan\left(\frac{v_y}{v_x}\right) \tag{3.3}$$

A velocidade é uma função da proximidade dos obstáculos, da diferença angular entre a direção desejada e a direção atual e da velocidade definida no passo anterior. A velocidade é dada por:

$$v_t = \eta\hat{v}_t + (1-\eta)v_{t-1} \tag{3.4}$$

com v_{t-1} a velocidade no instante $t-1$, $\hat{v}_t$ o valor estimado para v no instante t e η um fator de suavidade na transição das velocidades calculadas. O valor de $\hat{v}_t$ é uma função diferença entre a ângulo θ, da postura corrente do robô, e ângulo desejado ϕ. Isto é dado por $\varphi = \theta - \phi$, onde φ assume o menor arco entre θ e ϕ, ou seja, $-180^o < \varphi < 180^o$.

Além disso, define-se uma região de colisão de raio r_{max} centrada no robô. Se algum objeto for detectado a uma distância $d \leq r_{max}$, a velocidade sofre uma influência $\rho = \frac{d}{r_{max}}$, com d a distância ao obstáculo mais próximo. Assim, $\hat{v}_t$ é dado por:

$$\hat{v}_t = v_{max}\left(\frac{180-|\varphi|}{180}\rho\right) \tag{3.5}$$

com $\rho = 1$ se o robô estiver fora da região de colisão.

A versão clássica utilizando a equação de Laplace possui baixo desempenho em ambientes esparsos. Um ambiente *denso* é definido como aquele em que na maior parte do tempo existe algum obstáculo no campo de visão dos sensores do robô, ao contrário de ambientes *esparsos*. Ao se aproximar de uma parede, o robô é afastado antes que ele possa explorar melhor a região. Ao se aproximar da parede oposta, o mesmo fenômeno ocorre gerando um caminho oscilatório no qual o robô revisita um mesmo lugar diversas vezes, como veremos no capítulo 4.

Uma maneira de contornar este problema é a utilização de um conjunto mais genérico de funções, também geradoras de campos potenciais livres de mínimos locais. Inicialmente, a fim de quebrar a simetria do potencial e reduzir o efeito oscilatório em ambientes esparsos, substitui-se a equação de Laplace pela equação 3.6. Nesta equação, $\mathbf{v} \in \mathbb{R}^2$, com $|\mathbf{v}| = \mathbf{1}$, representa uma direção preferencial que independe da presença de obstáculos (PRESTES, 2003). O termo $\varepsilon \in \mathbb{R}$ representa a intensidade da perturbação.

$$\nabla^2 p(\mathbf{r}) = \varepsilon \mathbf{v} \cdot \nabla p(\mathbf{r}) \tag{3.6}$$

Posteriormente, foi proposta uma nova versão, dada pela equação 3.7, que supõe que $\mathbf{v}(\mathbf{r})$ é paralelo à $\nabla p(\mathbf{r})$. Esta nova equação permite um maior controle sobre a exploração, pois se regulando o termo $\varepsilon(\mathbf{r})$, definem-se regiões de alta e baixa preferência originadas pela deformação local do potencial (PRESTES; ENGEL, 2011).

$$\nabla^2 p(\mathbf{r}) = \varepsilon(\mathbf{r}) |\nabla p(\mathbf{r})| \tag{3.7}$$

Nos experimentos não são consideradas as equações (3.6) e (3.7) por estarem fora do escopo do plano de trabalho original.

3.2 Exploração utilizando Sensor-based Random Tree

O método SRT (ORIOLO et al., 2004) se baseia na construção incremental e aleatória de uma árvore que representa o roteiro da exploração com uma região segura associada. A região segura é o equivalente do conceito de região livre definido pela grade de ocupação, enquanto o resto, obstáculos e regiões não exploradas, pertencem a uma região não segura. Este algoritmo se diferencia por não fazer distinção entre obstáculos e regiões não exploradas, não empregando o conceito de fronteira.

Este método é probabilístico e inspirado em algoritmos como o Rapidly-exploring Random Trees (RRT) (LAVALLE; KUFFNER, 2001). Ele tem como vantagem a simplicidade e o fato de que qualquer sequência de ações será executada eventualmente, implicando que se alguma solução existir, esta será encontrada se o tempo necessário for fornecido. A seguir, o algoritmo geral é descrito, seguido de algumas extensões propostas.

3.2.1 Descrição do algoritmo

O método utiliza as seguintes suposições: (I) o ambiente é planar, i.e., $\mathbb{R}^2$ ou um subconjunto conectado de $\mathbb{R}^2$; (II) o robô é um disco livre para se deslocar em qualquer direção (holonômico); (III) o robô sempre sabe sua postura q; (IV) para cada q, o sistema sensorial é capaz de determinar o espaço livre ao redor do robô, definindo uma região segura local $\mathcal{S}(q)$, a qual deve possuir um formato de estrela. O algoritmo da Figura 3.5 descreve o funcionamento geral.

```
   Entrada: q_inicial, K_max, I_max, α, d_min
   Saída: T
 1 início
 2     q_atual = q_inicial ;
 3     para k = 1 até K_max faça
 4         S(q_atual) = PERCEPÇÃO( q_atual) ;
 5         ADICIONA( T,(q_atual, S(q_atual))) ;
 6         i = 0 ;
 7         repita
 8             θ_rand = DIREÇÃO_ALEATÓRIA;
 9             r = RAIO( S(q_atual), θ_rand) ;
10             q_candidato = DESLOCAR( q_atual, θ_rand, α · r) ;
11             i = i + 1 ;
12         até VALIDAR( q_candidato, d_min, T) ou i = I_max ;
13         se VALIDAR( q_candidato, d_min, T) então
14             MOVER_PARA( q_candidato) ;
15             q_atual = q_candidato ;
16         senão
17             MOVER_PARA( q_atual.pai) ;
18             q_atual = q_atual.pai ;
19         fim
20     fim
21     retorna T ;
22 fim
```

Figura 3.5: Algoritmo para exploração de ambientes utilizando SRT

A cada iteração k o robô percebe seu ambiente (linha 4) e define uma Região Segura Local (RSL) $\mathcal{S}$. Após, adiciona na árvore $\mathcal{T}$ um novo nodo, formado pelo par RSL e postura, como filho do nodo corrente (linha 5). No loop da linha 7, uma direção aleatória θ_{rand} é escolhida. A função RAIO retorna a distância até a borda de $\mathcal{S}$ na direção θ_{rand}, enquanto que o fator α, com $0 < \alpha < 1$, é utilizado para que $q_{candidato}$ sempre caia dentro da RSL. Se esta for válida, o robô avança para ela (linha 14), caso contrário, uma nova postura $q_{candidato}$ é escolhida, até I_{max} tentativas. Se nenhuma posição válida foi encontrada, o robô retrocede para a posição anterior na árvore (linha 17). Para uma posição candidata ser válida, esta deve: (a) estar a uma distância no mínimo d_{min} da posição atual e (b) não estar contida em nenhuma RSL de um nodo de $\mathcal{T}$. A Figura 3.6 ilustra a geração de posições candidatas. Note que a imposição (a) faz com que o robô se afaste naturalmente dos obstáculos, pois as posturas candidatas próximas a obstáculos estarão a uma distância menor que d_{min}, como a postura q''_{cand} vista na Figura 3.6. Já a imposição (b) faz com que o robô retorne ao explorar completamente o espaço livre, pois quando todo ambiente for explorado, todas as posturas candidatas estarão contidas em outras

RSL, como a postura q'_{cand} na Figura 3.6. Ao terminar a exploração, o robô se encontrará na posição $q_{inicial}$, caracterizando um mecanismo automático de regresso ao inicio. O mapa resultante é a união de todas RSL, as quais definem o espaço acessível ao robô.

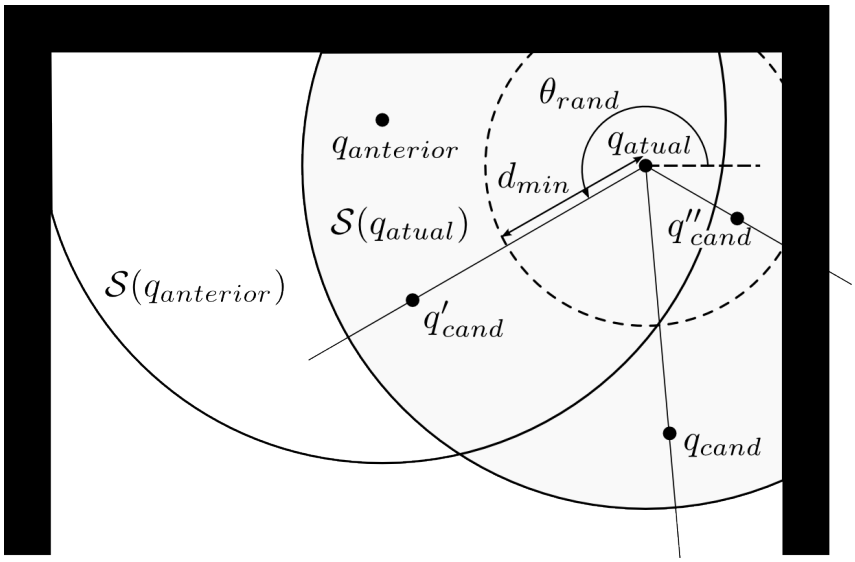
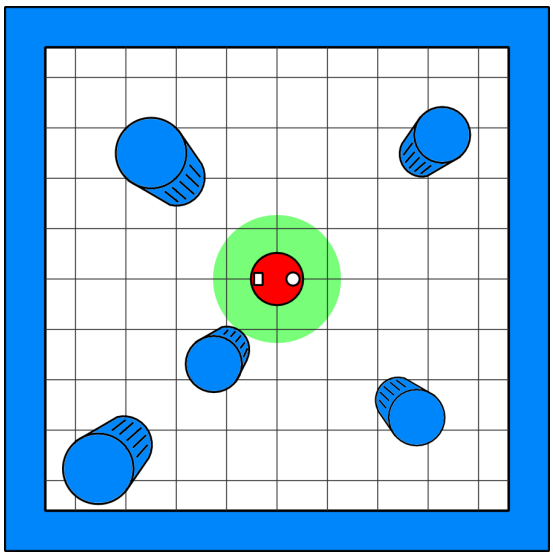


Figura 3.6: Critérios para validação de posições candidatas.

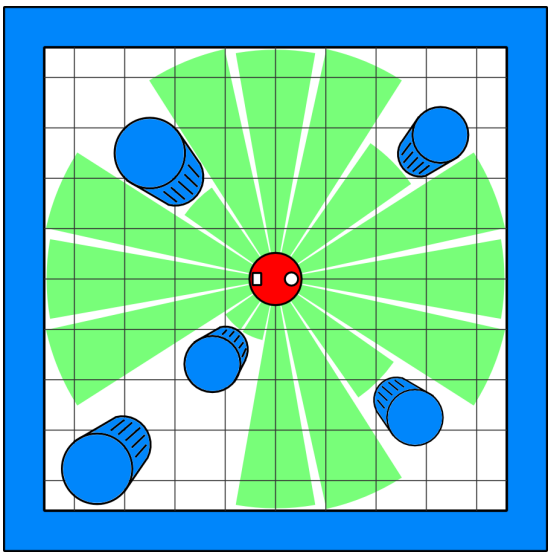
Este algoritmo geral depende fortemente da função PERCEPÇÃO utilizada, a qual deve gerar uma região em forma de estrela e depende do hardware disponível no robô. A Figura 3.7 apresenta três tipos diferentes de percepção dadas as leituras dos sensores. Na Figura 3.7a, utiliza-se uma abordagem conservativa chamada SRT-Bola, que consiste em definir a RSL como um disco com raio igual a menor distância obtida pelos sensores, sendo mais apropriada para sensores ruidosos. Uma versão mais confiante é apresentada na Figura 3.7b. Batizada SRT-Estrela, ela consiste em definir a RSL como um conjunto de cones centrados no robô devido ao uso de sensores do tipo sonar. Cada cone possui como raio a distância mínima dos sensores participantes do cone. Finalmente, utilizando-se sensores com uma alta densidade como sensores laser pode-se definir uma RSL como a da Figura 3.7c. Referida como SRT-Laser[1], a técnica define a RSL por um polígono cujos vértices são os pontos retornados pelo sensor.

Quanto maior a precisão da RSL em reproduzir o espaço livre, menor será a distância média percorrida, pois será necessário um menor número de nodos para visitar um mesmo ambiente. Além disso, essa precisão permite produzir um mapa mais fiel do ambiente explorado.

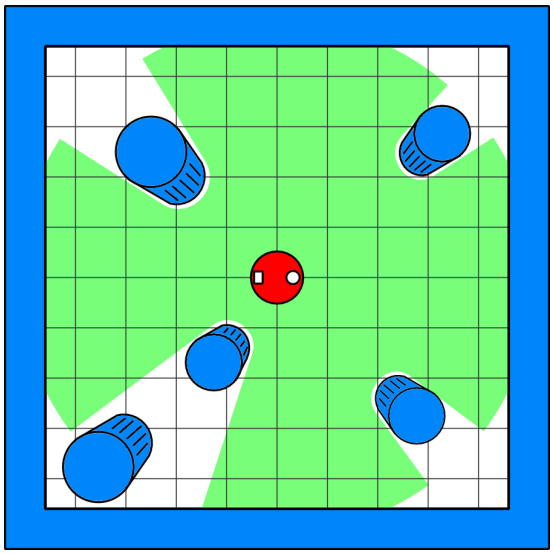
[1]Nos primeiros artigos, apenas as versões SRT-Ball e SRT-Estrela foram apresentadas. A partir da versão **integrada** (FREDA et al., 2006), utilizou-se apenas a versão baseada em sensores laser. Esta não foi batizada pelos autores, pois as outras técnicas ja não eram mais comentadas. Neste trabalho ela será chamada SRT-Laser para diferenciá-la.



(a) Região Segura Local para SRT-Bola



(b) Região Segura Local para SRT-Estrela



(c) Região Segura Local para SRT-Laser

Figura 3.7: Diferentes modos de percepção para o algoritmo SRT.

Figura extraída de (ORIOLO et al., 2004).

3.2.2 Extensões do algoritmo

Como dito anteriormente, o algoritmo original não faz distinção entre obstáculos e fronteiras. Assim, as posições candidatas são geradas uniformemente ao redor do robô e são necessárias diversas iterações até que uma posição candidata seja validada. A fim de aperfeiçoar o processo, Freda e Oriolo (2005) propuseram dividir o limite da RSL em arcos classificados como *fronteira*, *obstáculo* e *livre*. De posse dessas informações, o robô é capaz de influenciar a geração aleatória de posturas candidatas em direção a fronteiras, agilizando o processo de exploração. Além disso, a partir do número de fronteiras disponíveis é possível determinar

de forma direta se a região foi completamente explorada. Na versão original, uma região era completamente explorada se fossem geradas I_{max} posições candidatas inválidas (linha 12 do algoritmo da Figura 3.5).

Outras extensões, que vão além da *exploração clássica* e por isso não serão detalhadas, foram propostas: uma solução para o problema da *exploração integrada* (FREDA et al., 2006) e uma solução para o problema da *exploração coordenada utilizando múltiplos robôs* (FRANCHI et al., 2007).

4 EXPERIMENTOS

Neste capítulo serão apresentados os experimentos realizados para comparação dos dois algoritmos descritos no capítulo 3. Para os experimentos, a plataforma utilizada foi o robô Pioneer P3-DX da Adept Mobile Robots. É um robô diferencial com duas rodas, equipado com sensores do tipo sonar e laser com uma cobertura de 180^o. Possui uma interface de programação em C++, chamada **ARIA** (*Advanced Robot Interface for Applications*) , que permite recuperar os dados, como leitura dos sensores e odometria, e controlar a movimentação do robô. Além disso, também está disponível um simulador gráfico, chamado **MobileSim**, utilizado para validação e análises mais exaustivas dos algoritmos.

4.1 Algoritmos implementados

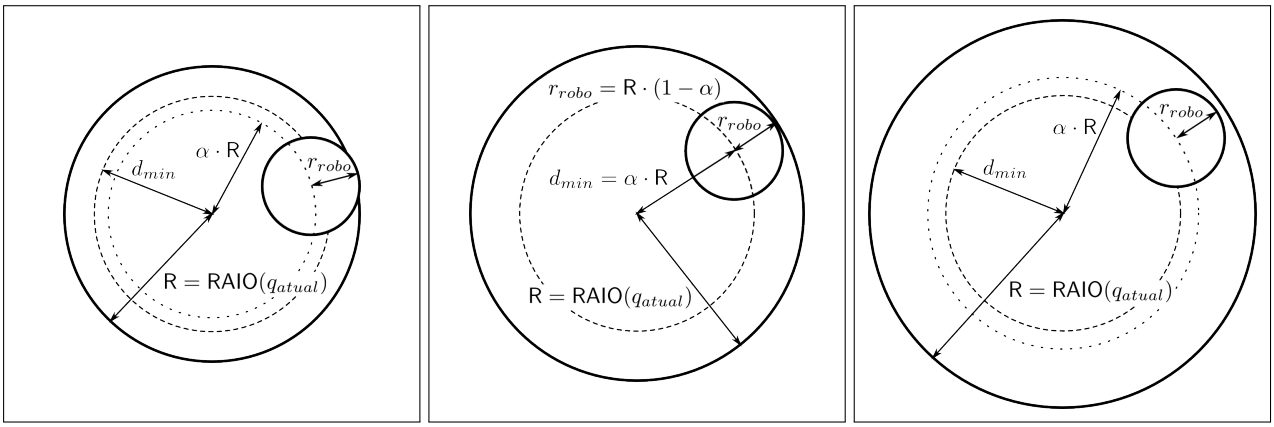
Para imlementação dos algoritmos descritos no capítulo 3 algumas escolhas precisaram ser feitas. Por exemplo: para o algoritmo SRT, o tipo de percepção e os parâmetros I_{max}, α e d_{min}; para o algoritmo PVC, o método de atualização da probabilidade de ocupação das células. Esta seção descreve alguns detalhes de implementação considerados importantes para compreensão dos resultados dos experimentos.

4.1.1 Sensor-based Random Tree

Como dito na seção 3.2, o algoritmo SRT depende fortemente do método de percepção escolhido. Para os experimentos, dois métodos foram implementados: SRT-Bola e SRT-Laser. O primeiro por sua simplicidade, baixo custo computacional e baixo consumo de memória: para cada nodo gerado, é preciso guardar apenas sua posição global, o valor do raio da RSL, e ponteiros para o nodo pai e nodos filhos.

Para o algoritmo SRT-Bola, deve-se determinar os valores de d_{min} e α. A condição fundamental é que o robô fique sempre dentro da RSL. Assim, o valor de d_{min} deve ser a mínima distância que satisfaça essa condição. Estes valores estão relacionados pelo raio do robô, r_{robo},

conforme demonstrado pela Figura 4.1. Se $\alpha \cdot \mathsf{R} < d_{min}$ o robô acaba fora da RSL (Figura 4.1a) e o robô pode colidir. O caso limite é $\alpha \cdot \mathsf{R} = d_{min}$ (Figura 4.1b). Neste caso, utilizando a equação $r_{robo} = \mathsf{R} \cdot (1 - \alpha)$, obtém-se a relação $d_{min} \cdot \left({}^{1}/_{\alpha} - 1\right) = r_{robo}$, a qual deve ser satisfeita. Assim, existem uma infinidade de pares (d_{min}, α) que são válidos para um dado r_{robo}. O valor de α regula o tamanho médio das arestas da árvore SRT. Quanto maior for α, mais o robô se aproxima da borda da RSL, maior é a probabilidade de ganho de informação, os nodos serão mais esparsos e será preciso um número menor de nodos para cobrir uma mesma região. No entanto, quanto maior for α, maior será d_{min}, o que determina o raio mínimo que a RSL deve ter para que o robô prossiga a exploração, impedindo o robô de atravessar corredores muito estreitos. Para ambientes densos, poder-se-ia utilizar um valor pequeno de α, de forma a diminuir d_{min}. Por exemplo, para um $\alpha = 0.66$ tem-se que $d_{min} = 2 \cdot r_{robo}$, implicando que o robô seria capaz de atravessar corredores tão estreitos quanto seu diâmetro. O problema de utilizar um α pequeno é que as RSL ficam muito próximas uma das outras, fazendo com que as posições candidatas geradas caiam sempre dentro de alguma outra RSL, e acabe não sendo validada. Assim, conclui-se que independentemente do parâmetros escolhidos, o algoritmo SRT-Bola não é adequado para ambientes densos. Como a penalidade induzida por um α pequeno faz o algoritmo falhar mesmo em ambientes esparsos, escolheu-se $\alpha = 0.8$. Como $r_{robo} = 250mm$, tem-se que $d_{min} = 1000mm$. Finalmente, determinou-se empiricamente que $I_{max} = 200$ era um valor suficiente para que o ambiente fosse corretamente explorado.



(a) $\alpha \cdot \mathsf{R} < d_{min}$ (b) $\alpha \cdot \mathsf{R} = d_{min}$ (c) $\alpha \cdot \mathsf{R} > d_{min}$

Figura 4.1: Determinação dos parâmetros para o algoritmo SRT-Bola.

Para o algoritmo SRT-Laser foi utilizada a extensão que considera as fronteiras na geração das posições candidatas. Deste modo, não é necessário ultrapassar I_{max} tentativas para determinar que não há mais o que explorar. Se não há fronteiras no nodo corrente, significa que a região foi toda explorada e robô retrocede para o nodo pai. No caso do SRT-Laser, não se pode definir uma posição candidata como uma fração do RAIO na direção de uma dada fronteira; pois, em

alguns casos, o caminho até este ponto causaria colisões em um robô não pontual. A Figura 4.2 apresenta uma RSL com fronteiras (linhas vermelhas) que demonstram este problema. A solução encontrada foi realizar um encolhimento da RSL para determinação dos pontos candidatos (pontos verdes na Figura 4.2). Cada ponto da fronteira é mapeado para a versão encolhida da RSL. Alguns pontos não conseguem ser mapeados, como visto no canto inferior direito da RSL, pois neste lugar a RSL é mais estreita que o dobro do offset desejado. Esta é uma propriedade interessante, pois permite a detecção de passagens muito estreitas que causariam a saída do robô da RSL. Assim, também não é preciso determinar valores para d_{min} e α.

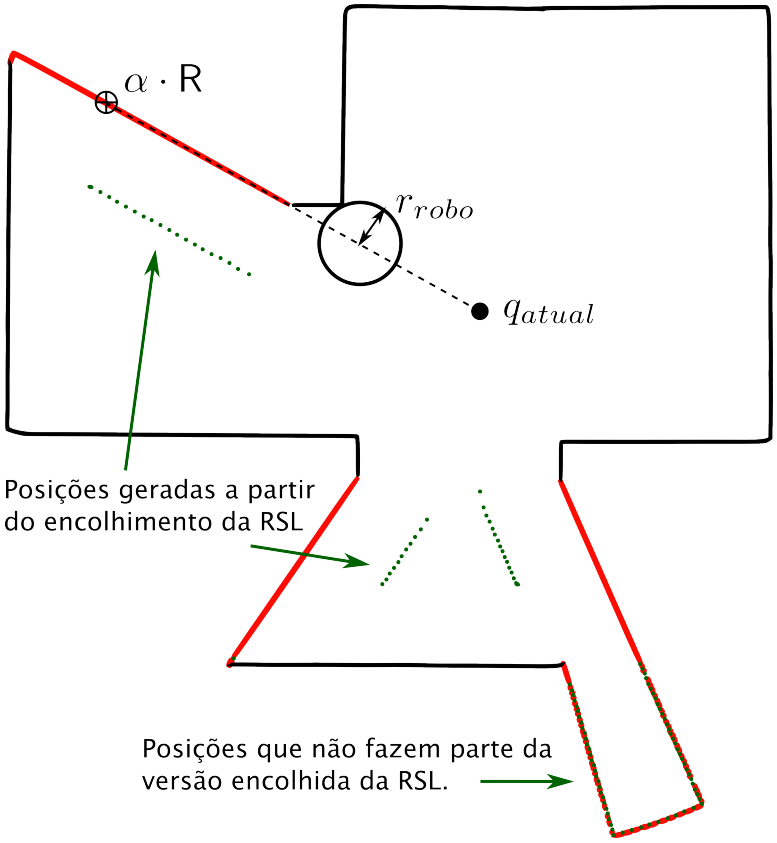


Figura 4.2: Determinação das posições candidatas para SRT-Laser.

4.1.2 Problemas de Valor de Contorno

O algoritmo baseado em PVC possuía mais detalhes de implementação na sua descrição que o SRT e estes foram seguidos sempre que possível. Para a atualização da probabilidade de ocupação das células, utilizou-se o método HIMM descrito anteriormente. A movimentação seguiu o modelo descrito na seção 3.1.2, com um raio da região de colisão $r_{max} = 200\,mm$. Para o cálculo do potencial, utilizou-se Gauss-Seidel, com um intervalo de 1 segundo entre cada atualização e 30 iterações por atualização. Vale ressaltar que a versão implementada utiliza funções harmônicas, o que é um caso particular da equação (3.7) com $\varepsilon(\mathbf{r}) = 0\ \forall \mathbf{r}$.

4.2 Interface para testar algoritmos de exploração

Para facilitar a realização dos experimentos e análise dos resultados foi implementada uma interface chamada de IfTEA (*Interface for Testing Exploration Algorithms*) . Ela possui três objetivos: (1) prover uma interface gráfica para gerenciar os parâmetros dos experimentos, (2) rodar em sistemas Windows e Linux e (3) prover uma interface de abstração para que os códigos dos algoritmos independam da plataforma física. Para cumprir os dois primeiros objetivos, foi utilizado o framework Qt da Nokia, que permite implementar interfaces gráficas para Windows, Linux e Mac OS X. Para atender a terceira condição, todos os algoritmos utilizam uma classe abstrata para acessar funcionalidades do robô. Assim, apenas a classe que implementa a interface da classe abstrata faz chamadas às funções específicas fornecidas pela biblioteca ARIA.

4.2.1 Funcionalidades

A Figura 4.3 mostra a interface gráfica implementada para configuração, monitoração e coleta de resultados dos experimentos. Inicialmente, apenas o painel de configuração está disponível, permitindo que diversos parâmetros sejam configurados antes de começar a exploração. Em relação ao robô, pode-se determinar seu *endereço*, o *alcance máximo do laser* e sua *velocidade máxima*. Estipula-se a taxa de redesenho do mapa e opcionalmente uma pasta onde serão salvas imagens do mapa regularmente, permitindo a posterior criação de um vídeo. Definem-se os limites do mapa em milímetros e o método de exploração. Dependendo do método escolhido, opções adicionais podem ser informadas, como tamanho da célula em métodos que utilizem grades de ocupação e o intervalo entre atualizações do potencial para o PVC.

Os limites do mapa são importantes para algoritmos que utilizam grades de ocupação pois a memória necessária para representar o mapa é alocada uma única vez no início da exploração e depende do número de total de células. Assim, se o ambiente a ser explorado for maior que o tamanho determinado de antemão, a exploração falhará. Esta limitação foi aceita pois introduzir alocação dinâmica de matrizes acarretaria um aumento na complexidade dos algoritmos de exploração que está fora do escopo deste trabalho. Por sua vez, o algoritmo SRT aloca dinamicamente a árvore que representa seu mapa e o algoritmo não falhará independentemente das medidas fornecidas. No entanto, como estas são utilizadas para renderizar o mapa na tela, a aplicação não conseguirá desenhar além dos limites informados, comprometendo a visualização.

Após iniciada a exploração, o menu **Display** oferece funcionalidades para o monitoramento

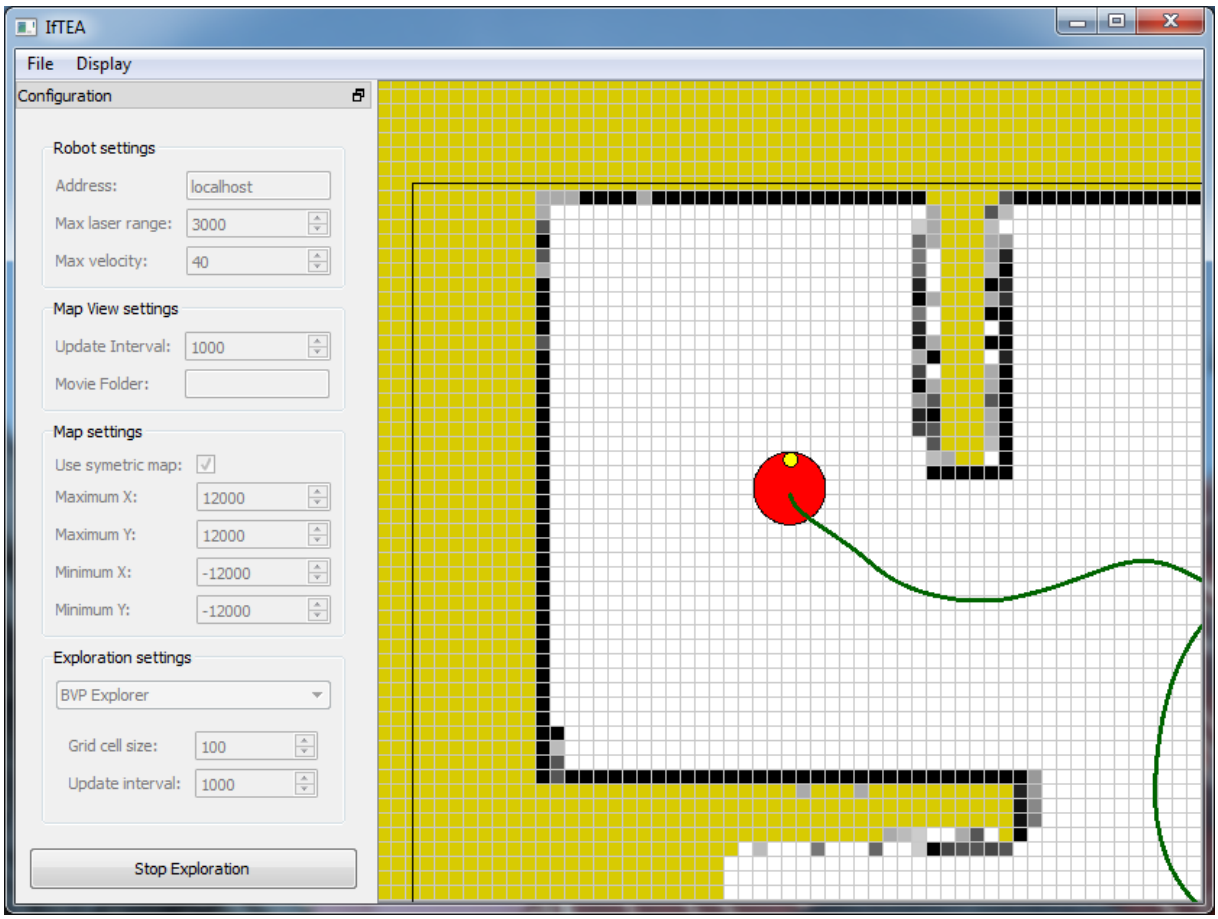
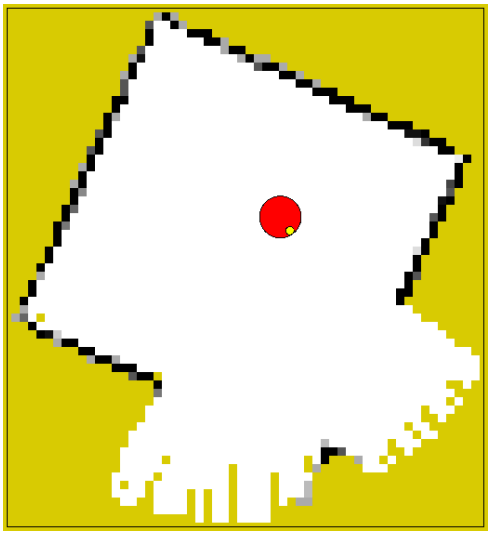


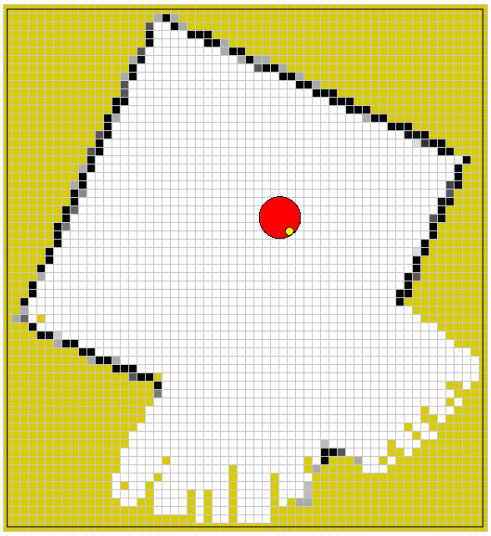
Figura 4.3: Interface gráfica implementada para realização dos experimentos.

do processo exploratório realizado pelo robô. Independentemente do método de exploração pode-se: mostrar o caminho percorrido pelo robô; mostrar as leituras laser; habilitar o *tracking* do robô para mantê-lo sempre no centro; modificar a escala para desenho do mapa através das funções **Zoom in** e **Zoom out**; salvar o mapa como uma imagem no formato PNG. Além disso, a exploração utilizando PVC fornece as seguintes opções, que podem ser combinadas de qualquer maneira: mostrar grade (Figura 4.4b), mostrar campo vetorial (Figura 4.4c) e mostrar campo potencial (Figura 4.4d).

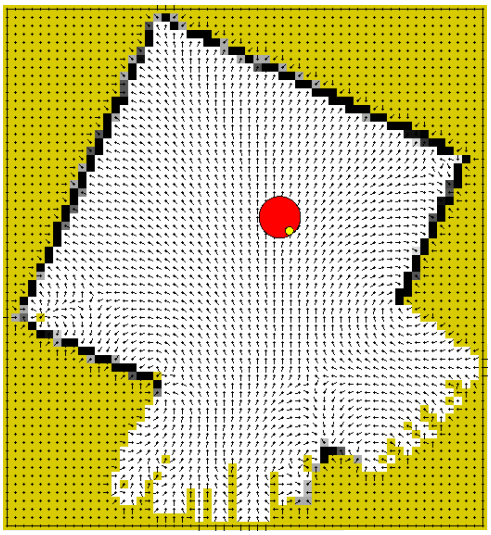
Depois de concluída a exploração, uma mensagem aparece informando o tempo em milissegundos necessário para a exploração e o comprimento percorrido pelo robô em milímetros. Todas opções de visualização continuam habilitadas, e o mapa resultante pode ser salvo através da opção **Save map as image**. A edição das configurações da exploração voltam ser habilitadas, permitindo que uma nova exploração seja realizada.



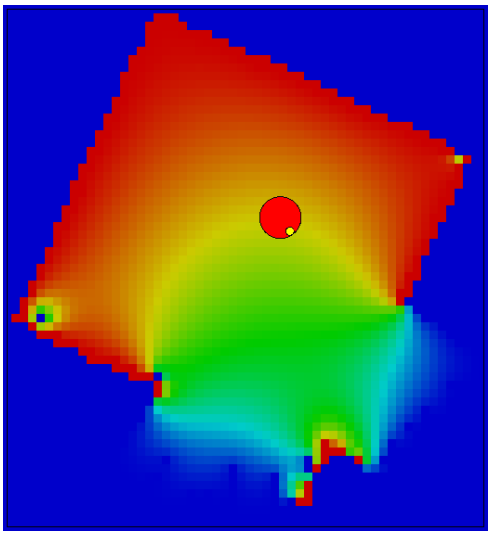
(a) Nenhuma opção habilitada



(b) Opção mostrar grade



(c) Opção mostrar campo vetorial



(d) Opção mostrar campo potencial

Figura 4.4: Diferentes opções para visualização de mapas para exploração utilizando PVC.

4.2.2 Automação em Lua

Apesar de a interface apresentada permitir que diferentes experimentos fossem executados sem a necessidade de recompilar o programa, as configurações e coleta dos resultados precisavam ser feitas manualmente. Executar diversos experimentos em sequência exigia a intervenção manual a cada quarenta minutos (tempo médio de exploração) tornando o processo muito lento e difícil. Para resolver este problema, implementou-se a automatização da interface através de scripts Lua.

A opção **Load Lua script** no menu **File** permite carregar um script Lua. Quando um script está executando, as mensagens de final de exploração são omitidas. Além de ter acesso à todas configurações e opções de visualização, também é possível deslocar o robô no simulador

MobileSim e esperar pelo final da exploração. Como exemplo, no código abaixo o robô será posicionado para cada postura definida em *DensePoses* no MobileSim, inicia-se a exploração e, ao terminar, os resultados são escritos em um arquivo e uma imagem do mapa é salva.

```
fout = io.open([[D:\UFRGS\TCC\auto\resultados.txt]], "a+");

DensePoses = {
  [1] = {x = -5000, y = 7500, theta = 0 },
  [2] = {x = -1600, y = 7500, theta = 0 },
  [3] = {x = 2700, y = 7500, theta = 0 }
}

function results (name, n)
  elapsedTime = exploration.elapsedTime();
  path = robot.pathLength();
  fout:write([[
  Map: ]] .. name .. [[ Pose: ]] .. n .. [[
  Elapse time: ]] .. elapsedTime .. [[
  Path length: ]] .. path .. [[
  ]]);
  fout:flush();
  mapView.setMapScaling(0.1);
  robotView.setShowPath(true);
  mapView.setShowGrid(true);
  mapView.saveMapImage([[D:\UFRGS\TCC\auto\]]..name..[[_]]..n..[[.PNG]]);
end

robot.setAddress("localhost");
robot.setMaxLaserRange(3000);
exploration.setMethod(ExplorationMethod.BVP);

for num, pose in pairs(DensePoses) do
  fout:write("Starting BVP in pose " .. num .. "\n");
  fout:flush();
  robot.moveMobileSim(pose.x, pose.y, pose.theta);
  exploration.start();
  exploration.wait();
  results("Denso_PVC", num);
end
```

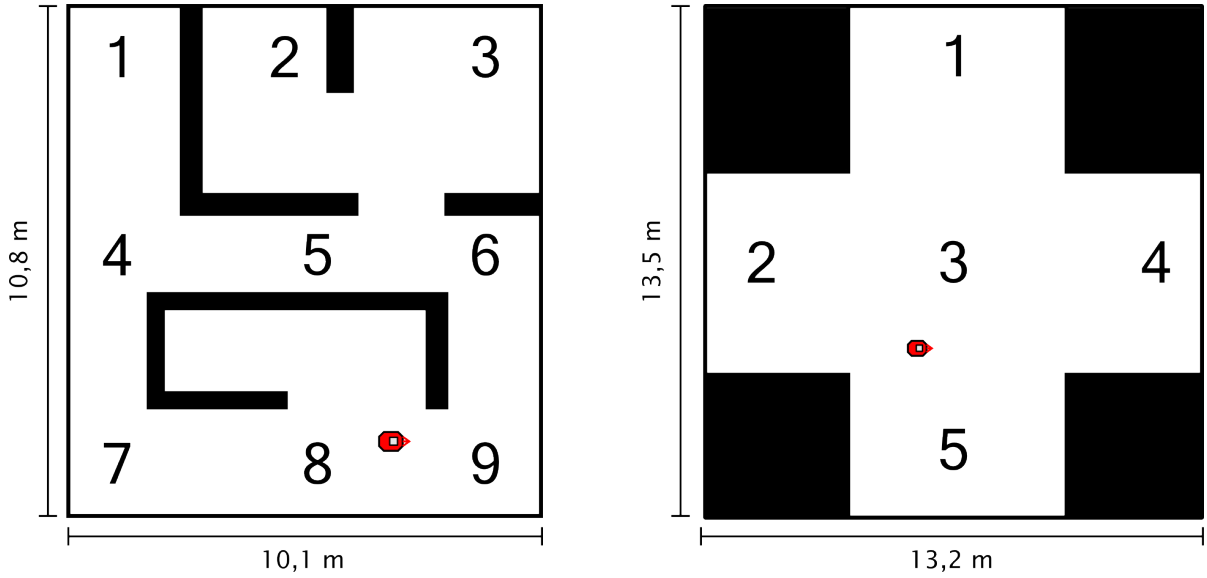
4.3 Ambientes

Para avaliação dos algoritmos de exploração foram realizadas simulações e experimentos. As simulações utilizam o simulador MobileSim, que permite carregar diferentes mapas, simular sensores do tipo sonar e laser, além de modelar erros de odometria.

Durante todos os testes, os seguintes parâmetros foram utilizados: alcance do laser de 3000 mm e tamanho da célula para o PVC de 100 mm. Foi sempre especificado um tamanho máximo para o mapa grande o suficiente, ou seja, maior que as medidas do ambiente a ser explorado. A seguir são descritos os ambientes utilizados e a descrição dos testes realizados.

4.3.1 Simulação

As simulações foram feitas utilizando um ambiente denso e um ambiente esparso. Para o ambiente denso foram definidos 9 pontos de partida, representados pelos números 1 a 9 na Figura 4.5a, enquanto para o ambiente esparso foram definidos 5 pontos de partida, representados pelos números 1 a 5 na Figura 4.5b. Todas as simulações foram executadas utilizando duas velocidades máximas: 100 mm/s e 40 mm/s. Como observado na seção 4.1.1, o algoritmo SRT-Bola não é adequando para ambientes densos. Assim, para o ambiente denso foram executados os algoritmos PVC e SRT-Laser, enquanto para o ambiente esparso os algoritmos PVC, SRT-Laser e SRT-Bola.



(a) Mapa denso (b) Mapa esparso

Figura 4.5: Ambientes utilizados para simulação.

4.3.2 Experimentos

Foram realizadas mais simulações, pois os experimentos utilizando o robô Pioneer P3-DX eram limitados por três fatores: uso de bateiras, necessidade de supervisionar o mesmo durante todo processo de exploração e erros de odometria muito grandes. Além disto, a velocidade máxima do robô simulado pode ser maior que a velocidade máxima do robô real. O maior fonte dos erros de odometria eram erros sistemáticos: as rodas possuem tamanhos diferentes fazendo com que ele fizesse uma curva quando solicitado para seguir em linha reta. O sistema de odometria é gerenciado pela biblioteca ARIA, que não permite configurar o tamanho das rodas.

Para resolver o problema, duas soluções podem ser consideradas: reimplementar o sistema de odometria ou utilizar um algoritmo de SLAM. Ambas as soluções necessitariam de um esforço de pesquisa e implementação muito grande, e fogem do escopo do trabalho desenvolvido. Assim, os experimentos foram utilizados para analisar os algoritmos de forma mais qualitativa do que quantitativa, pois os tempos de execução e caminho percorrido fazem menos sentido como será visto na seção 4.4.2. De todo modo, alguns experimentos foram realizados para comparar os algoritmos de forma qualitativa e ressaltar as diferenças entre eles. A Figura 4.6 apresenta o ambiente utilizado para os experimentos no robô real: uma pequena sala, composta por mesas de trabalho, cadeiras e um pequeno armário.

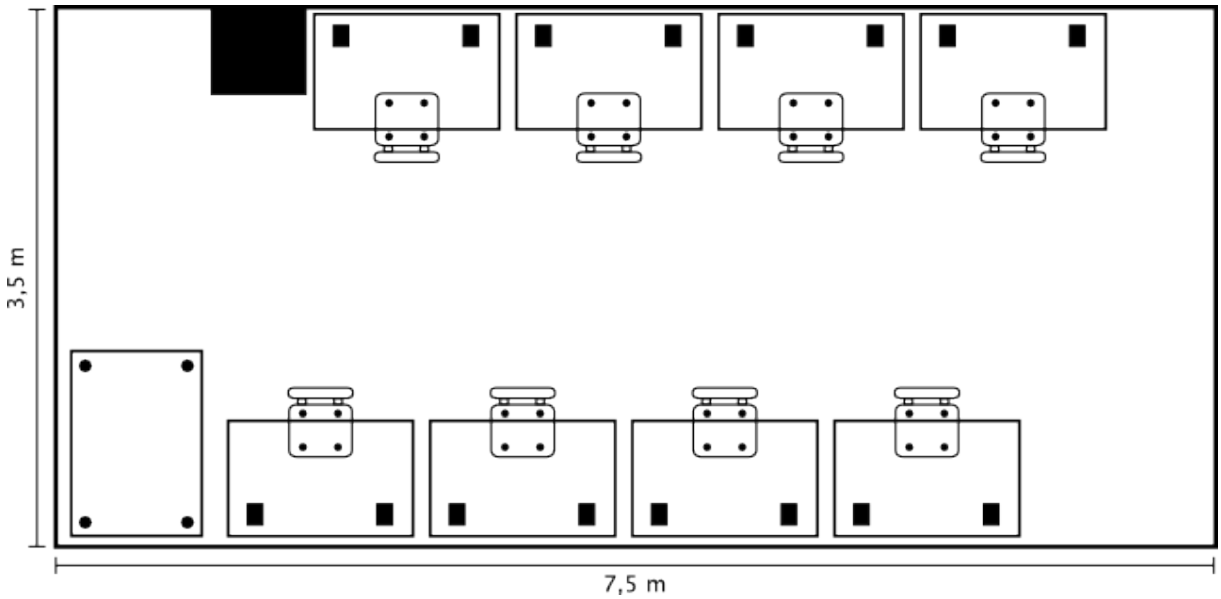


Figura 4.6: Ambiente utilizado para experimentos no robô real.

4.4 Resultados

Nesta seção são detalhados os resultados das simulações e experimentos. O grande número de simulações permitiu uma análise quantitativa dos algoritmos em termos de tempo de execução e caminho percorrido. Vale relembrar que estes índices de *qualidade* fazem sentido apenas na exploração clássica, uma vez que a exploração integrada também considera a fidelidade do mapa produzido, exigindo revisitas e consequente aumento do tempo de exploração e caminho percorrido.

4.4.1 Ambientes Simulados

Para compreensão dos resultados, alguns pontos precisam ser considerados: (1) O algoritmo SRT faz com que o robô sempre volte ao ponto de partida da exploração. Isto ocorre naturalmente pois ele retrocede ao nodo pai quando nenhuma postura candidata é validada; (2) a natureza do potencial utilizado pelo PVC faz com que algumas vezes o robô prefira fronteiras grandes mais distantes que fronteiras pequenas próximas (ver Figura 3.3). Esse comportamento de *desistência* das fronteiras muito pequenas faz com que ele precise revisitar o local posteriormente para completar a exploração. Nos mapas resultantes, o caminho realizado pelo robô é visto em verde no caso do PVC. Para o SRT, como o caminho coincide com as arestas, este foi omitido. Note que o robô sempre atravessa cada aresta exatamente duas vezes: uma quando ele está chegando à nova posição e outra quando ele está retornando ao nodo pai.

Ambiente denso e velocidade máxima de 100 mm/s

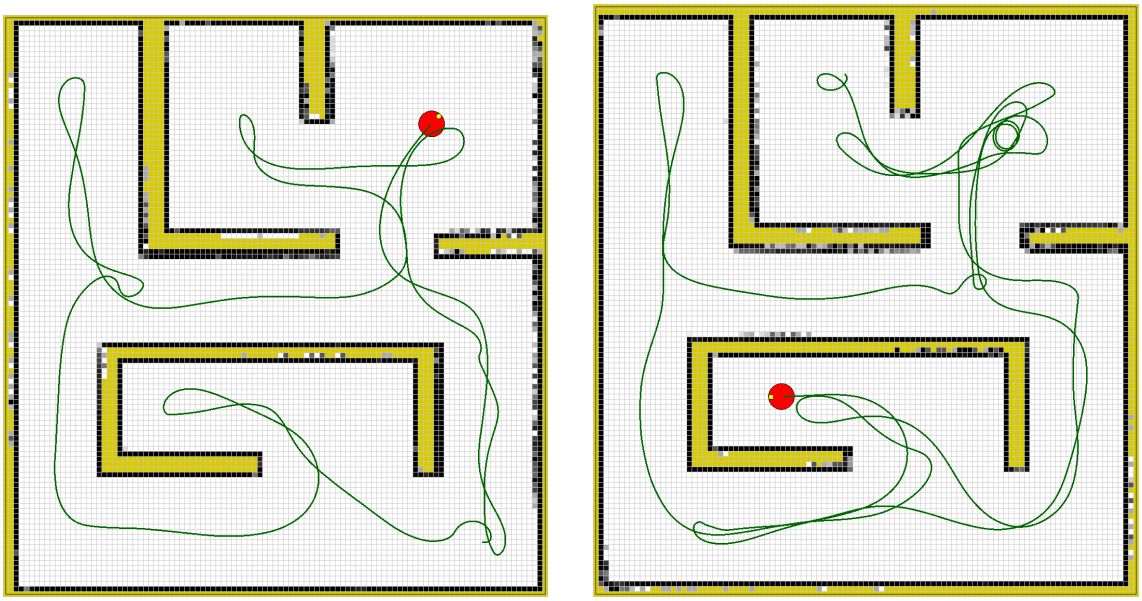
A tabela 4.1 apresenta os resultados para as simulações utilizando o mapa denso (Figura 4.5a) e velocidade máxima de 100 mm/s. Apesar de o algoritmo SRT possuir um fator randômico associado à exploração, é o PVC que sofre maior influência da posição inicial e maior disparidade de resultados. A amplitude total (diferença entre o melhor e pior resultado) em termos de caminho percorrido para o PVC foi de 25 m enquanto que para o SRT-Laser de 7,8 m. Como visto na tabela 4.1, o caminho percorrido pelo SRT-Laser está fortemente relacionado ao número total de nodos necessários para representar o ambiente. Como o número de nodos depende pouco da posição inicial, o caminho e o tempo de execução variam pouco com a posição inicial. O pior caso para o SRT-Laser foi o único a precisar de 19 nodos, visto na Figura 4.7d. O melhor caso, Figura 4.7c, precisou apenas de 17 nodos, mas como visto no canto inferior esquerdo, uma pequena porção do ambiente não foi mapeada. O algoritmo baseado em PVC sofreu bastante com a desistência das pequenas fronteiras e diversas regiões foram revisitadas no pior caso, como visto na Figura 4.7b e mesmo no melhor caso, Figura 4.7a.

Utilizando uma velocidade máxima de 100 mm/s o algoritmo SRT-Laser mostrou-se mais eficiente mesmo com o sobre custo de retornar à posição inicial. Apesar de ambos possuírem um tempo de execução médio muito próximo (16 minutos e 30 segundos), o SRT-Laser possui um caminho 7,9 metros mais curto em média que o PVC. Um ponto positivo do PVC foi a geração de caminhos suaves. Esta é uma característica importante pois este tipo de caminho produz menos erros de odometria, gerando um caminho com menos incerteza e mais fácil de ser corrigido por um algoritmo de SLAM. Contrariamente, uma grande fonte de incerteza é o movimento de rotação, bastante presente no algoritmo SRT-Laser, que dificulta o trabalho do

Tabela 4.1: Simulação utilizando ambiente denso e velocidade máxima de 100 mm/s

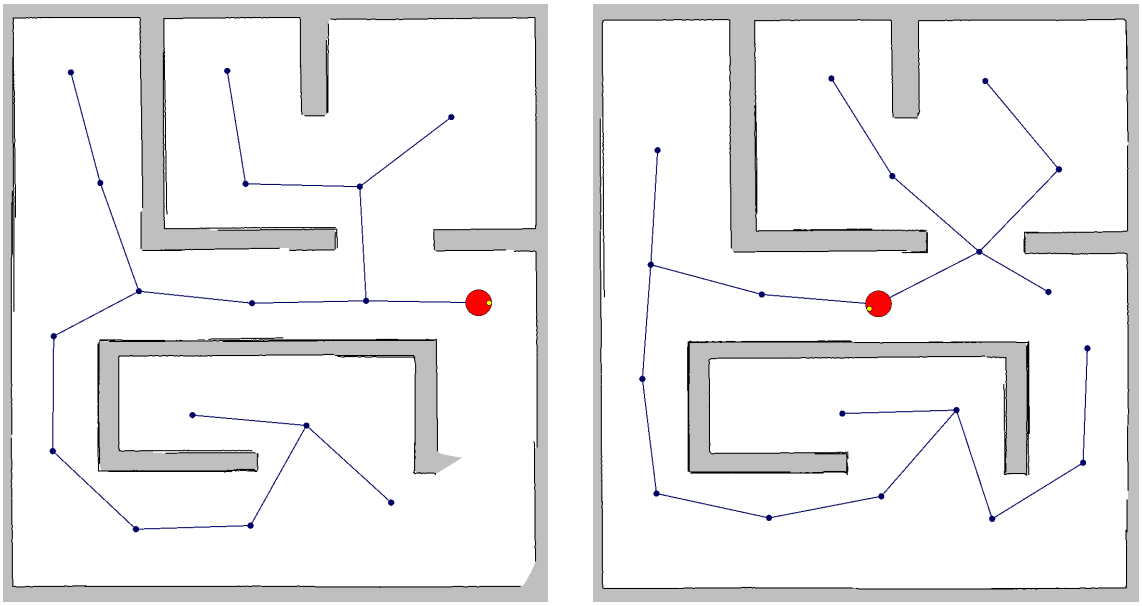
	PVC (100 mm/s)		**SRT-Laser** (100 mm/s)		
Postura	Caminho (m)	Duração (min)	Caminho (m)	Duração (min)	Nodos
1	74,5	15:21	70,7	16:43	18
2	93,9	19:52	74,4	17:09	18
3	85,5	17:47	69,8	16:15	17
4	79,7	16:52	69,7	15:67	17
5	89,0	18:42	77,0	17:32	19
6	69,3	14:06	69,2	15:42	17
7	74,3	14:56	69,8	16:13	17
8	84,1	17:27	70,0	15:57	17
9	68,3	14:06	73,7	17:03	18
Média:	79,8	16:34	71,6	16:39	18
Desvio padrão:	8,9	02:04	2,7	00:53	1
Amplitude:	25,5	05:46	7,8	01:50	2
Vel. média:	80 mm/s		72 mm/s		

SLAM.



(a) Melhor PVC: postura 9

(b) Pior PVC: postura 2



(c) Melhor SRT-Laser: postura 6

(d) Pior SRT-Laser: postura 5

Figura 4.7: Mapas da simulação com ambiente denso e velocidade máxima de 100 mm/s.

Ambiente denso e velocidade máxima de 40 mm/s

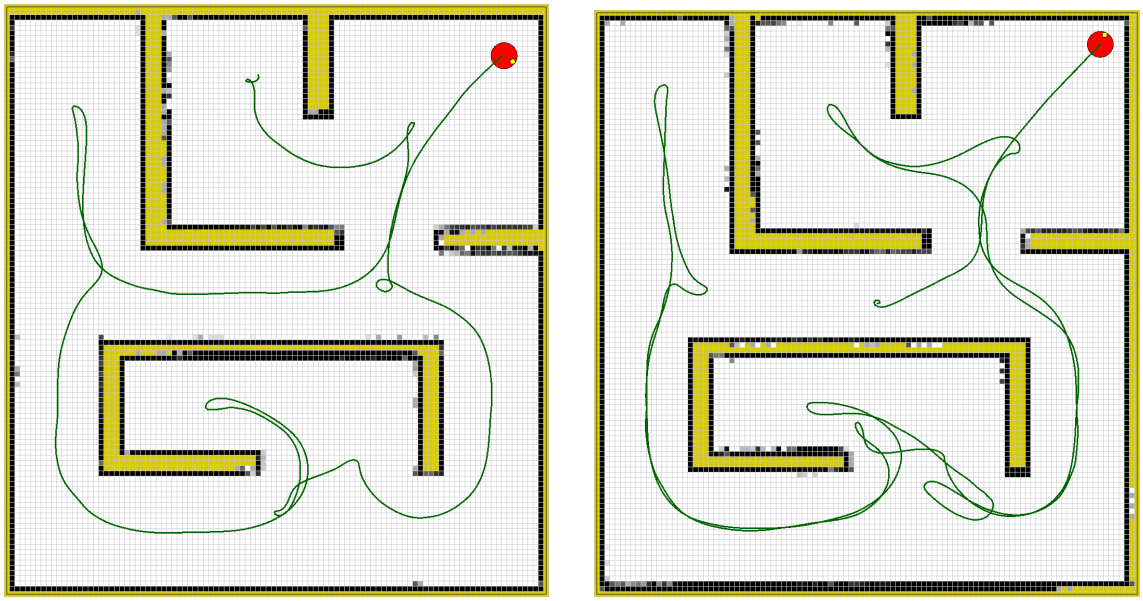
O próximo conjunto de simulações foi realizado com o intuito de observar a influência da velocidade máxima no desempenho dos algoritmos. Os resultados encontram-se na tabela 4.2. O algoritmo baseado em PVC continua sendo o mais suscetível à posição inicial, com uma amplitude total de 23,4 metros para o caminho percorrido, enquanto o SRT-Laser teve sua amplitude acrescida a 13,1 metros. O aumento na diferença entre o melhor e pior caso do SRT-Laser está relacionado com a execução do mapa em apenas 16 nodos partindo da postura 4 (Figura

4.8c). Note como neste caso o mapa encontra-se incompleto próximo ao centro do mapa. Isto ocorreu pois as fronteiras eram muito pequenas (do ponto de vista local) e o algoritmo decidiu não visitá-las.

De modo geral, o algoritmo SRT-Laser manteve um caminho médio percorrido igual ao caso anterior e um tempo de execução proporcionalmente maior. Já o algoritmo baseado em PVC teve uma redução considerável no caminho médio percorrido, passando de 79,8 para 61,7 metros. Note como o robô quase não revisita o mapa no melhor caso (Figura 4.8a) e mesmo no pior caso (Figura 4.8b) é melhor que o caminho médio à velocidade máxima de 100 mm/s. Este fenômeno está relacionado com método de mapeamento utilizado. Ao passar mais lentamente em uma região, as células acabam recebendo mais atualizações em um mesmo intervalo de tempo. Como o ambiente é melhor mapeado, formam-se menos "buracos", que precisariam ser revisitadas posteriormente. Assim, a 40 mm/s, o melhor algoritmo é o baseado em PVC, com um caminho 10,3 metros menor em média e um tempo de execução de 7 minutos e 21 segundos menor em média do que o SRT-Laser. Além disso, como discutido anteriormente, o caminho realizado pelo PVC é mais suave e portanto mais adaptado para um algoritmo de SLAM.

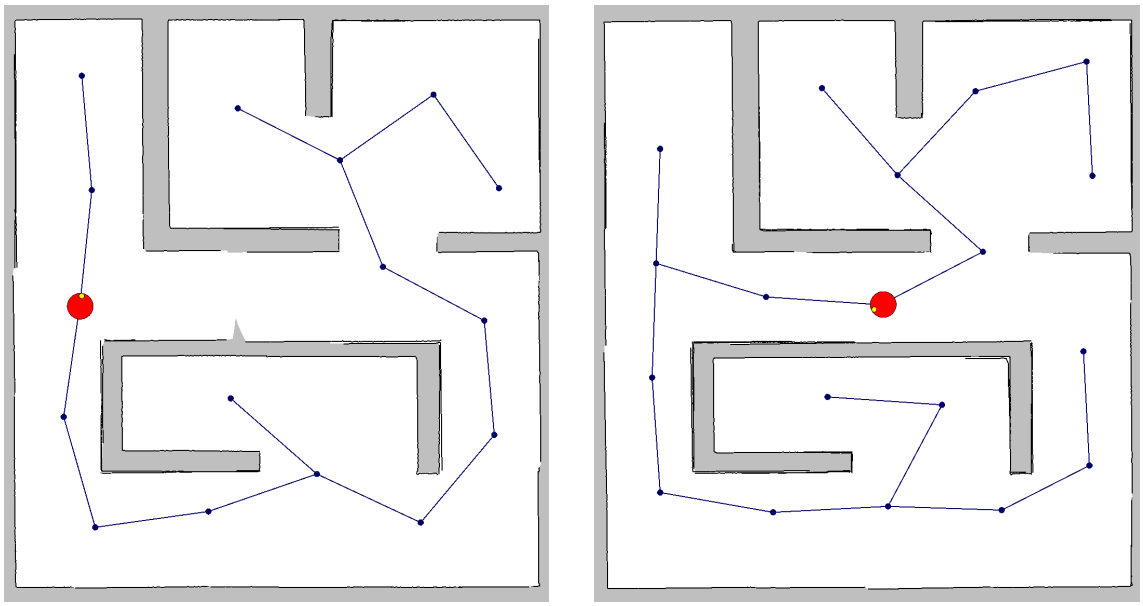
Tabela 4.2: Simulação utilizando ambiente denso e velocidade máxima de 40 mm/s

	PVC (40 mm/s)		**SRT-Laser** (40 mm/s)		
Postura	Caminho (m)	Duração (min)	Caminho (m)	Duração (min)	Nodos
1	63,4	28:13	68,5	31:47	17
2	51,5	23:42	78,2	36:53	19
3	65,6	29:13	68,3	31:58	17
4	60,9	26:53	65,1	30:03	16
5	74,5	32:43	77,5	39:24	19
6	51,0	22:17	68,8	34:48	17
7	60,0	25:53	73,9	34:12	18
8	55,8	24:22	74,1	34:36	18
9	72,0	31:23	73,3	37:05	18
Média:	61,7	27:11	72,0	34:32	18
Desvio padrão:	8,3	03:31	4,5	02:58	1
Amplitude:	23,4	10:26	13,1	09:11	3
Vel. média:	38 mm/s		35 mm/s		



(a) Melhor PVC: postura 2

(b) Pior PVC: postura 5



(c) Melhor SRT-Laser: postura 4

(d) Pior SRT-Laser: postura 5

Figura 4.8: Mapas da simulação com ambiente denso e velocidade máxima de 40 mm/s.

Ambiente esparso e velocidade máxima de 100 mm/s

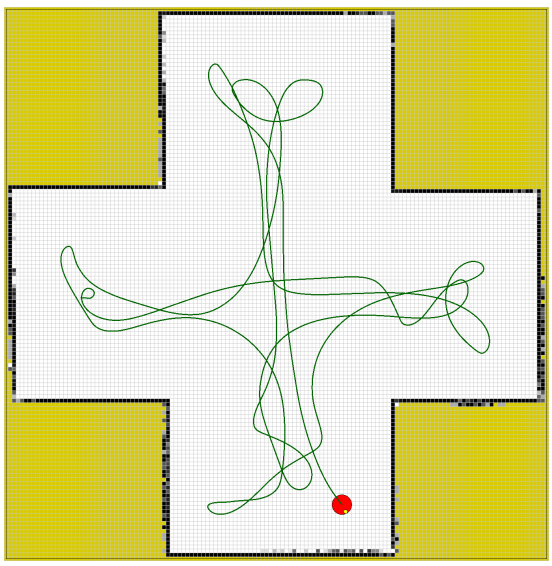
Utilizando o ambiente da Figura 4.5b e uma velocidade máxima de 100 mm/s, foram obtidos os dados da tabela 4.3. A diferença de performance entre os algoritmos PVC e SRT-Laser aumentou consideravelmente. O comportamento errático do robô pode ser visto no melhor e pior caso, Figuras 4.9a e 4.9b respectivamente. Com um caminho médio de 117 metros, o PVC precisou quase o dobro do SRT-Laser, que utilizou em média de 64,7 metros. Existem dois fatores que levam a esta diferença. O primeiro é derivado do uso de funções harmônicas no PVC.

Sabe-se que estas possuem um baixo desempenho em ambientes esparsos, devido à simetria do potencial gerado. Além disso, como todo método que maximiza a distância de obstáculos, o robô começa a se afastar das paredes antes de poder explorar adequadamente a região. O segundo vem do comportamento de busca em profundidade do SRT-Laser: o robô sempre avança na exploração e retrocede apenas quando não há mais nada à ser explorado localmente, minimizando as revisitas, como observado nas Figuras 4.9c e 4.9d.

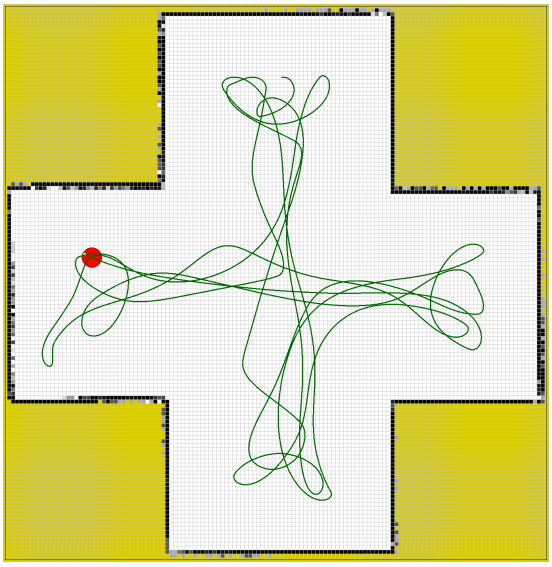
O SRT-Bola, que não pode ser executado no ambiente denso devido à problemática discutida na seção 4.1.1, foi executado no ambiente esparso. Ele obteve, como esperado, o pior desempenho em todos os aspectos. Um fato interessante que pode ser observado é que o pior caso no SRT-Bola (Figura 4.9f) tem menos nodos que o melhor caso (Figura 4.9e). Isto é possível pois no SRT-Bola existe maior variação na distância entre os nodos que no SRT-Laser. Assim, havendo poucos nodos, mas mais espaçados, é possível ter um caminho total maior e consequentemente maior tempo de execução.

Tabela 4.3: Simulação utilizando ambiente esparso e velocidade máxima de 100 mm/s

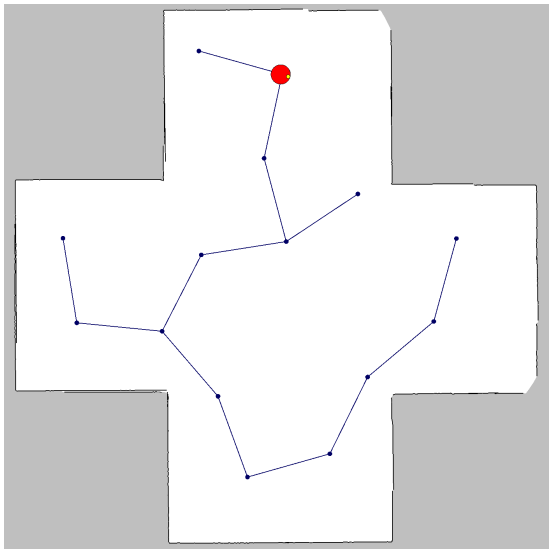
	PVC (100 mm/s)		**SRT-Laser** (100 mm/s)			**SRT-Bola** (100 mm/s)		
Postura	Caminho	Duração	Caminho	Duração	Nodos	Caminho	Duração	Nodos
1	139,6	25:37	61,2	14:21	15	190,1	49:50	65
2	99,5	18:22	74,3	17:47	18	179,2	47:20	66
3	123,7	22:27	61,3	14:30	15	179,8	46:44	63
4	113,0	20:47	65,6	15:17	16	183,7	47:51	68
5	109,3	19:57	61,2	14:21	15	188,2	47:36	64
Média:	117,0	21:26	64,7	15:15	16	184,2	47:52	65
Desvio:	15,3	02:46	5,7	01:28	1	4,9	01:10	2
Amplitude:	40,1	07:16	13,1	03:26	3	10,9	03:06	5
Vel. média:	91 mm/s		71 mm/s			64 mm/s		



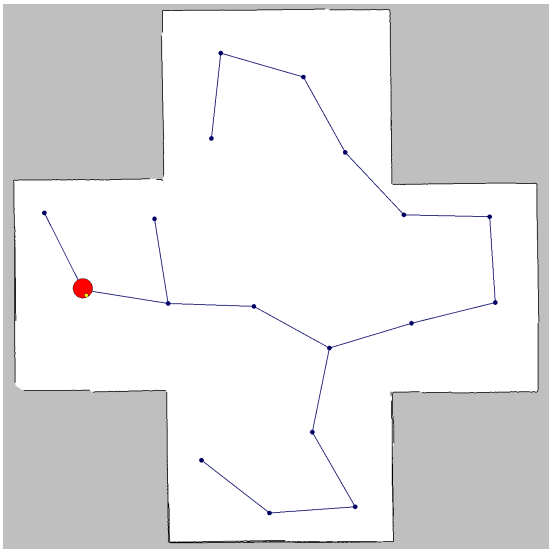
(a) Melhor PVC: postura 2



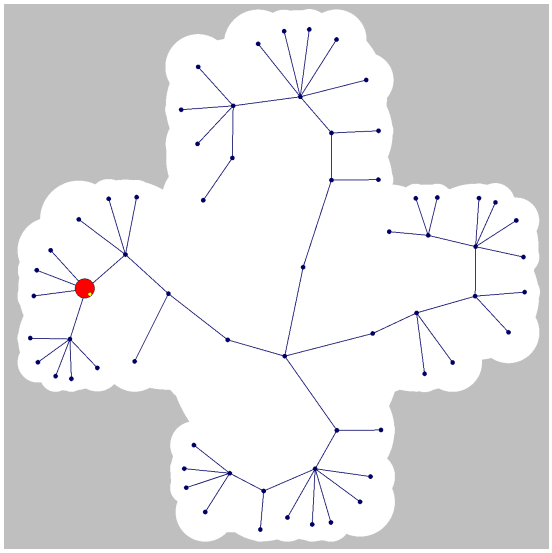
(b) Pior PVC: postura 1



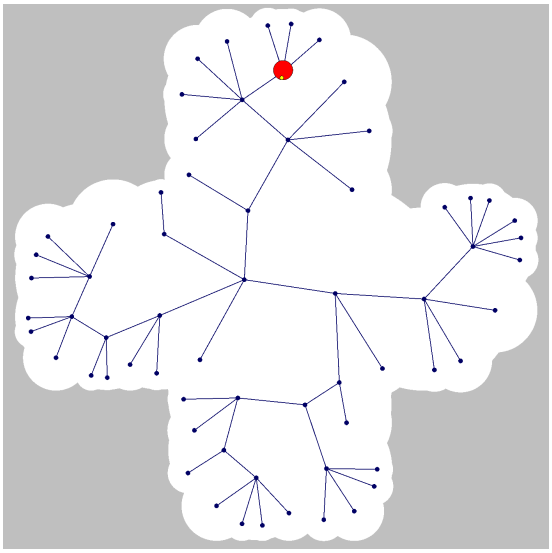
(c) Melhor SRT-Laser: postura 1



(d) Pior SRT-Laser: postura 2



(e) Melhor SRT-Bola: postura 2



(f) Pior SRT-Bola: postura 1

Figura 4.9: Mapas da simulação com ambiente esparso e velocidade máxima de 100 mm/s.

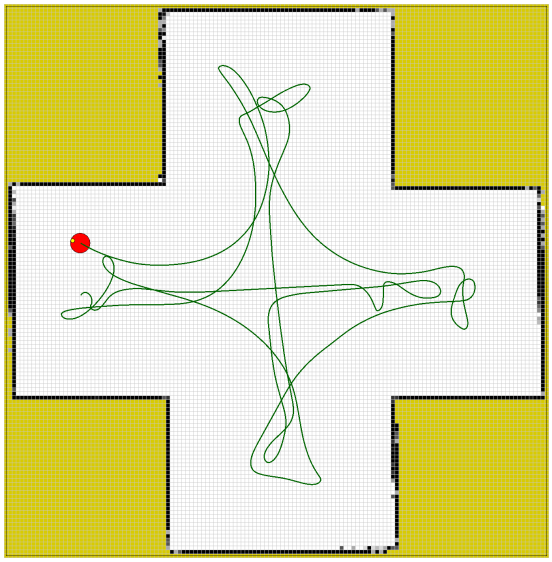
Ambiente esparso e velocidade máxima de 40 mm/s

Os resultados para a simulação no ambiente esparso com uma velocidade máxima de 40 mm/s encontram-se na tabela 4.4. Apesar do pequeno ganho em desempenho do algoritmo PVC, passando a um caminho médio de 100,9 metros, o algoritmo SRT-Laser continua muito melhor, com uma média de 69,5 metros. O comportamento do robô continua errático no PVC (Figura 4.10b) e mesmo o melhor caso (Figura 4.10a) obteve um pior desempenho que o pior caso caso do SRT-Laser (Figura 4.10d). O aumento para o SRT-Laser é devido ao caso excepcional que necessitou de 19 nodos. No melhor caso, foram utilizados apenas 16 nodos (Figura 4.10c), resultando em uma caminho total de apenas 64,9 metros.

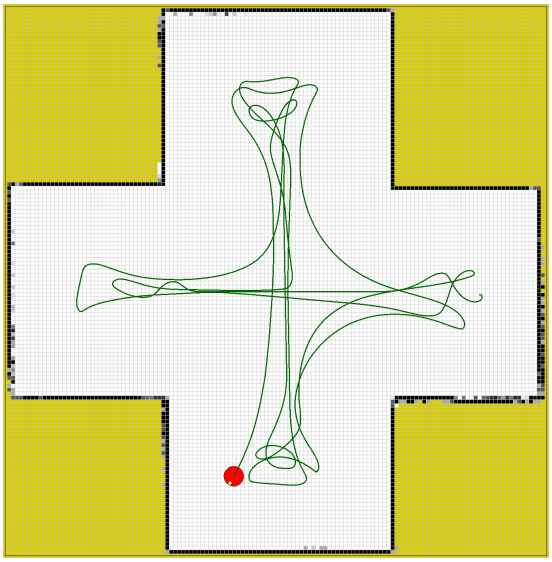
O SRT-Bola teve uma leve redução no caminho médio, mas continuou sendo muito pior que os outros dois algoritmos, com um tempo de execução médio três vezes maior que o SRT-Laser. Isto é consequência do número muito grande nodos necessários para mapear o ambiente, o que resulta em mais arestas, maior caminho e maior tempo de execução. É importante observar que o melhor caso para o SRT-Bola (Figura 4.10e) apresenta muitas regiões não exploradas, produzindo um mapa mais incompleto que o pior caso (Figura 4.10f). Assim, para o SRT-Bola uma análise apenas com base no caminho e tempo de execução não é apropriada para classificar como *melhor* e *pior* caso.

Tabela 4.4: Simulação utilizando ambiente esparso e velocidade máxima de 40 mm/s

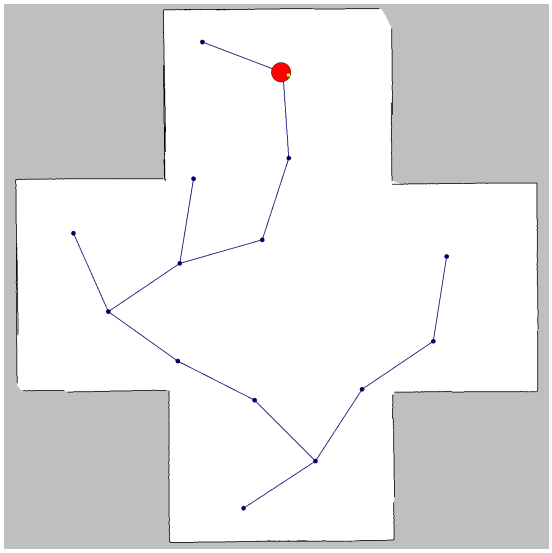
	PVC (40 mm/s)		**SRT-Laser** (40 mm/sec)			**SRT-Bola** (40 mm/sec)		
Postura	Caminho	Duração	Caminho	Duração	Nodos	Caminho	Duração	Nodos
1	105,4	46:29	64,9	30:26	16	184,7	1:31:45	65
2	84,5	36:39	75,6	36:15	19	171,3	1:24:00	58
3	106,0	45:14	69,6	32:56	17	162,8	1:21:03	59
4	106,1	45:39	67,8	32:16	17	198,2	1:39:49	76
5	102,3	44:44	69,5	32:41	17	170,0	1:25:54	65
Média:	100,9	43:45	69,5	32:55	17	177,4	1:28:30	65
Desvio:	9,3	04:02	3,9	02:06	1	14,1	07:26	7
Amplitude:	21,6	09:51	10,7	05:48	3	35,4	18:46	18
Vel. média:	38 mm/s		35 mm/s			33 mm/s		



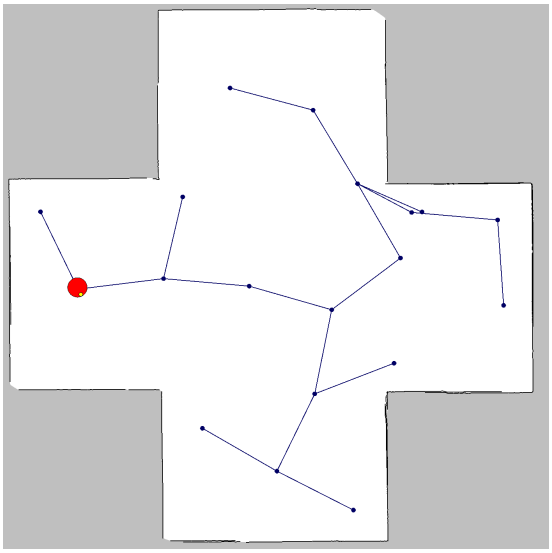
(a) Melhor PVC: postura 2



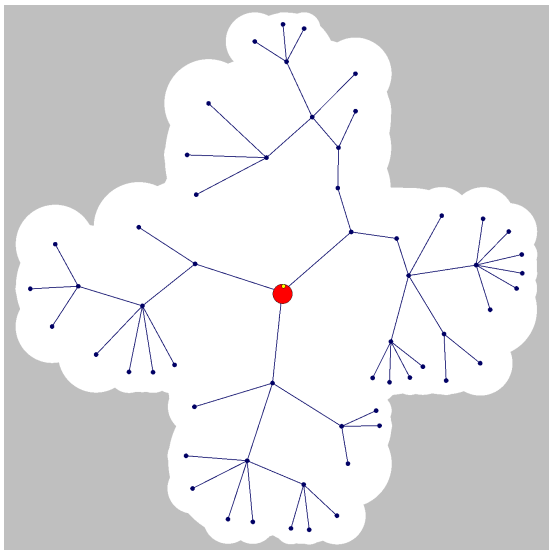
(b) Pior PVC: postura 4



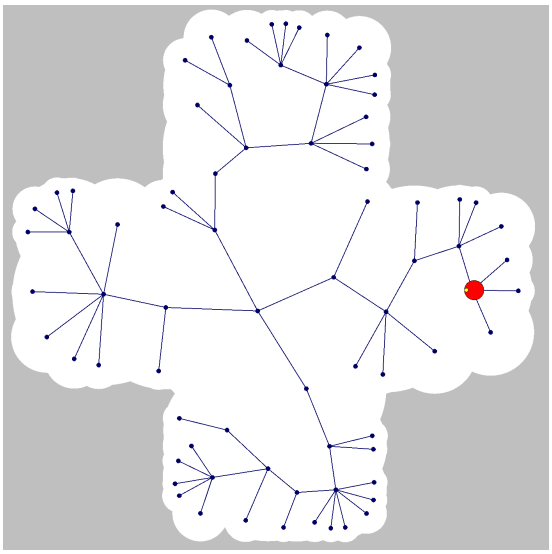
(c) Melhor SRT-Laser: postura 1



(d) Pior SRT-Laser: postura 2



(e) Melhor SRT-Bola: postura 3



(f) Pior SRT-Bola: postura 4

Figura 4.10: Mapas da simulação com ambiente esparso e velocidade máxima de 40 mm/s.

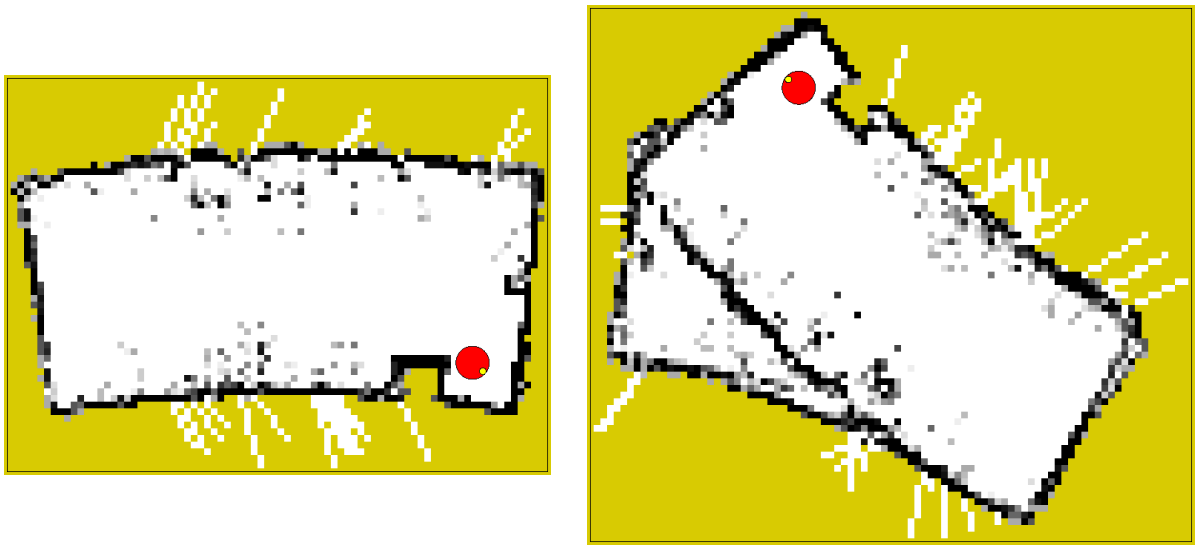
4.4.2 Experimentos em ambientes reais

A ambiente da Figura 4.6 apresentou dois desafios: os múltiplos finos pés das cadeiras e o canto atrás do armário. Enquanto o algoritmo SRT-Laser sofreu com os pés das cadeiras, o PVC foi penalizado pelo canto formado pela parede e o armário.

A Figura 4.11 mostra dois casos para a exploração utilizando PVC. No primeiro caso, Figura 4.11a, o robô consegue explorar todo ambiente sem a necessidade de revisitar regiões já exploradas. Já no segundo caso, Figura 4.11b, o robô precisa retornar para completar a exploração e o mapa resultante sofre maior deformação, consequência do efeito cumulativo dos erros de odometria. Essa necessidade de revisitar ocorre no segundo caso pois o robô começa próximo ao armário. Nesta situação, uma pequena fronteira se forma na região oculta pelo armário e sua força atratora é menor que a grande fronteira que o leva ao resto da sala. Assim, o robô vai primeiro explorar o resto da sala para finalmente voltar para pequena fronteira. É importante observar que em ambos os casos as pernas das cadeiras não impuseram nenhuma dificuldade para o algoritmo. Nos mapas da Figura 4.11, as cadeiras são os pontos espalhados no interior da sala. Eles não chegam a ocultar as paredes, mas são capazes de exercer uma força de repulsão suficientemente grande para que o robô não colida com elas e permaneça no meio da sala, como pode ser visto pelo caminho percorrido por ele.

Vale ressaltar que o algoritmo baseado em PVC foi capaz de concluir a exploração apenas por ser capaz de tratar com *ambientes dinâmicos*. Apesar de não ter efetivamente objetos móveis na sala, os erros de odometria faziam com que para o robô as paredes parecessem em movimento, uma vez que suas coordenadas absolutas se moviam conforme o robô avançava. Esta capacidade de lidar com ambientes dinâmicos depende mais da técnica de mapeamento do que do método de exploração. Note que estes dois não estão completamente acoplados. A única restrição é que o mapeamento deve gerar um mapa do tipo grades de ocupação, enquanto que a técnica utilizada para atualização da probabilidade de cada célula é indiferente para o método de exploração.

Contrariamente, o algoritmo SRT não é capaz de lidar com ambientes dinâmicos. A captura de informação ocorre apenas na geração do nodo. Se o ambiente mudar, não há como o nodo refletir esta mudança. Além disso, o método de movimentação simplificado também não é capaz de desviar de obstáculos. Como o robô deve sempre retornar à posição inicial, os erros de odometria foram tão importantes que na maioria dos casos o robô colidiu com os obstáculos. A Figura 4.12 apresenta um caso onde a exploração foi concluída com sucesso. Observe como a leitura corrente do laser, mostrada em azul, está desalinhada com o mapa gerado. Outro ponto importante que pode ser observado é a influência das pernas finas das cadeiras. A forma



(a) Sem revisitas (b) Com revisitas

Figura 4.11: Resultado dos experimentos para sala pequena utilizando PVC.

em estrela da RSL faz com que a parede fique oculta pelas pernas das cadeiras. Como as informações dos nodos não são atualizadas conforme o robô avança, o mapa não poderá ser completado e o resultado são os "buracos"observados na Figura 4.12.

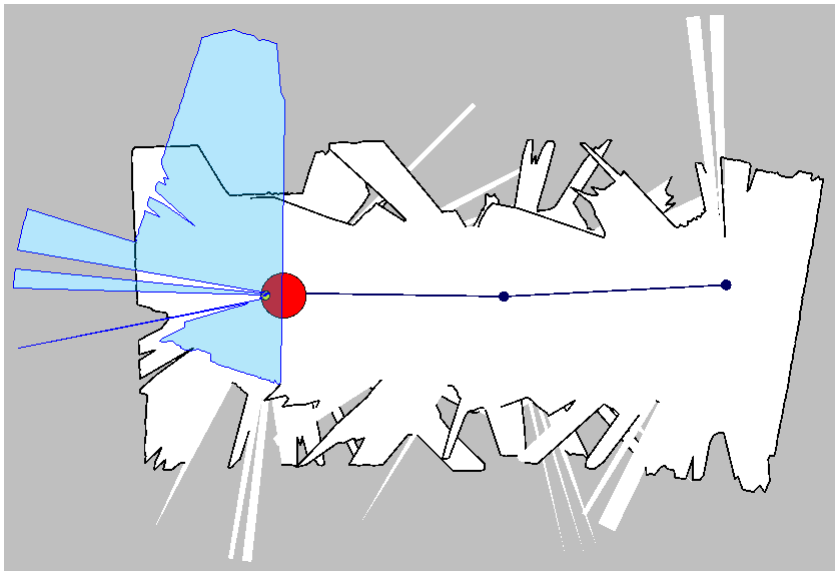
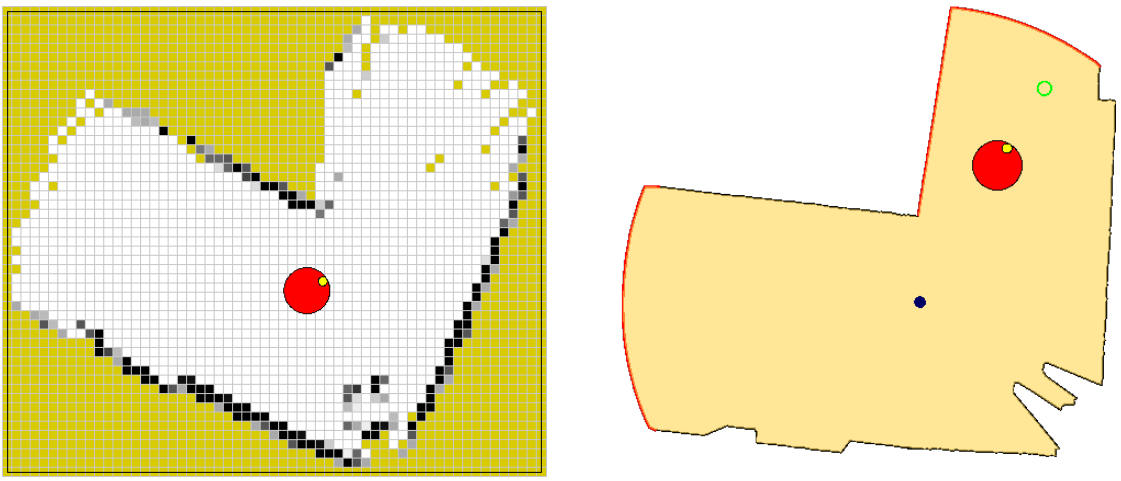


Figura 4.12: Resultado dos experimentos para sala pequena utilizando SRT-Laser.

A última análise é em relação à fidelidade do mapa gerado, considerando uma localização perfeita. A Figura 4.13 mostra o mapa intermediário obtido durante a exploração de um mesmo ambiente para os algoritmos PVC e SRT-Laser. No canto inferior direito de ambos os mapas, encontra-se uma pessoa sentada. A figura 4.13b apresenta maior resolução e detalhes que per-

mitiriam reconhecer que há uma pessoa sentada de forma mais fácil do que se utilizássemos apenas as informações disponíveis no mapa da Figura 4.13a. Em ambos os casos, o nível de detalhamento não está diretamente associado ao algoritmo de exploração. No caso do PVC, o detalhamento é determinado pelo tamanho da célula de ocupação. Utilizando-se uma célula pequena o suficente, o detalhamento do PVC poderia ser equivalente ao obtido pelo SRT-Laser. No entanto, seria computacionalmente impraticável calcular o campo potencial resultante. No caso do SRT, o detalhamento depende do tipo de RSL utilizada. Tanto o SRT-Estrela como o SRT-Bola não seriam melhores que o PVC. A grande vantagem do SRT é que ele continua computacionalmente vantajoso mesmo usando todo detalhamento permitido pelo laser.



(a) PVC (b) SRT-Laser

Figura 4.13: Comparação do detalhamento do mapa entre os algoritmos SRT-Laser e PVC.

5 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foram apresentados dois métodos de exploração de ambientes desconhecidos para robótica móvel. Com abordagens distintas, as simulações e experimentos mostraram que existem situações onde cada algoritmo é mais adequado.

A principal vantagem do algoritmo SRT vem de sua simplicidade, baixo custo computacional e baixo custo de memória. Foram implementadas duas versões: SRT-Bola e SRT-Laser. A primeira mostrou-se completamente inadequada para ambientes densos e em ambientes esparsos apresentou uma baixa eficiência em termos de caminho percorrido e tempo de execução, além de não ter sido capaz de explorar a totalidade do ambiente. Apesar de ter gerado mapas incompletos, uma porcentagem significativa do ambiente foi coberto e nunca ocorreram colisões. Com essas duas considerações, demonstrou-se que ela é uma solução válida e adequada para sistemas com requisitos de memória e processamento restritos e que aceitem uma cobertura parcial do ambiente. Além disso, um robô de raio muito pequeno seria capaz de explorar ambiente da Figura 4.5a, considerado denso, utilizando o SRT-Bola, pois para ele poder-se-ia definir um α e d_{min} adequados.

A versão SRT-Laser faz uso de toda informação disponível pelo sensor do tipo laser. Além disto, utilizou-se a otimização que considera as fronteiras, aumentando ainda mais a diferença de eficiência entre os dois métodos SRT. O SRT-Laser obteve resultados muito bons tanto em ambientes densos como esparsos. Além disso, estes resultados não foram influenciados pela velocidade máxima do robô. Porém, vale ressaltar que erros de odometria são proporcionais à velocidade; sendo assim, em uma aplicação real, a velocidade teria uma influência na *qualidade* do mapa gerado.

Em contrapartida, o PVC sofreu diretamente influência da velocidade máxima. Isto está relacionado com método de mapeamento empregado: quanto mais lentamente o robô avança, mais vezes as células são atualizadas em um dado intervalo de tempo, evitando assim pequenos "buracos"nas paredes e consequentes revisitas para "tapá-los". Um dos problemas da versão

implementada do PVC foi seu péssimo desempenho em ambientes esparsos. Apesar da diminuição da velocidade ter melhorado um pouco seu desempenho neste caso, o SRT-Laser mostrou-se muito mais efetivo que o PVC.

Muitos métodos de planejamento utilizam grades de ocupação para determinação das trajetórias. Isto dá ao PVC uma vantagem pois o mapa resultante pode ser diretamente utilizado por estes algoritmos enquanto o SRT-Laser precisaria ter sua representação interna convertida. No entanto, este tipo de mapa também dá uma desvantagem ao PVC. É necessário alocar toda memória que representará o mapa de antemão, o que implica que as dimensões máximas do mapa precisam ser informadas. O SRT-Laser, por sua vez, realiza a alocação de memória de forma dinâmica, uma vez que seu mapa é salvo em uma estrutura do tipo árvore.

Além disto, o custo computacional do PVC aumenta conforme o mapa aumenta, pois o potencial precisa ser computado em toda região conhecida. Assim, o PVC não é adequado para ambientes muito grandes. Por sua vez, o SRT faz uma análise local a cada iteração e seu custo computacional independe do número de nodos corrente ou do tamanho do mapa.

Apesar de todos os pontos positivos em relação ao SRT-Laser, este não é capaz de lidar com ambientes dinâmicos, pois a aquisição de dados ocorre apenas na geração dos nodos. Isto pode ser um fator decisivo dependendo da aplicação desejada, fazendo com que o PVC seja o mais adequado para uma toda uma classe problemas.

O trabalho apresentado pode ser estendido considerando-se as diversas melhorias propostas pelos autores dos algoritmos a fim de obter uma comparação mais justa. Por exemplo, espera-se que o PVC aumente seu desempenho em ambientes esparsos se forem utilizadas as melhorias descritas no final da seção 3.1.2. Também se poderia implementar e comparar as versões de exploração integrada, ou integrar novas abordagens para melhor comparação.

A exploração de ambientes desconhecidos é uma área da robótica em constante evolução e diversos algoritmos são propostos sem que se tenha tempo de compará-los e compreender seus pontos fortes e fracos. Apesar deste trabalho não ter introduzido um novo algoritmo de exploração, a análise apresentada é de extrema importância para uma melhor compreensão das opções existentes e base para avaliação de novas soluções.

REFERÊNCIAS BIBLIOGRÁFICAS

ALDEBARAN ROBOTICS. *Aldebaran Robotics*. 2005. Disponível em: < http://www.aldebaran-robotics.com/>. Acesso em: novembro 2011.

AURENHAMMER, F. Voronoi diagrams: a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, v. 23, n. 3, 1991.

BAILEY, T.; NEBOT, E. Localisation in large-scale environments. *Robotics and Autonomous Systems*, v. 37, p. 261–281, 2001.

BORENSTEIN, J.; KOREN, Y. Histogramic in-motion mapping for mobile robot obstacle avoidance. *IEEE Transactions on Robotics and Automation*, v. 7, p. 535–539, 1991.

BURGARD, W.; FOX, D.; THRUN, S. Active mobile robot localization. In: *Advanced Mobile Robots, 1999. Proceedings from the Second EUROMICRO workshop on*. [S.l.: s.n.], 1997.

CHOSET, H. et al. Exact cellular decompositions in terms of critical points of morse functions. In: *Robotics and Automation. ICRA 2000. Proceedings of the 2000 IEEE International Conference on*. [S.l.: s.n.], 2000. v. 3, p. 2270–2277.

CONNOLLY, C. I.; GRUPEN, R. A. On the applications of harmonic functions to robotics. *Journal of Robotic Systems*, v. 10, p. 931–946, 1993.

DELAVAL. *Ordenha Voluntária*. 1998. Disponível em: <http://www.delaval.com.br/-/Product-Information1/Milking/Systems/Voluntaria/> Acesso: novembro 2011.

ELFES, A. Using occupancy grids for mobile robot perception and navigation. *Computer*, p. 46–57, 1989.

FILLIAT, D.; MEYER, J.-A. Map-based navigation in mobile robots: I. a review of localization strategies. *Cognitive Systems Research*, v. 4, p. 243–282, 2003.

FORTUNA, A. de O. *Técnicas computacionais para dinâmica de fluidos: conceitos básicos e aplicações*. [S.l.]: Editora da Universidade de São Paulo, 2000.

FRANCHI, A. et al. A randomized strategy for cooperative robot exploration. In: *Robotics and Automation. ICRA 2007. Proceedings of the 2007 IEEE International Conference on*. [S.l.: s.n.], 2007.

FREDA, L.; LOIUDICE, F.; ORIOLO, G. A randomized method for integrated exploration. In: *Intelligent Robots and Systems. Proceedings of the 2006 IEEE/RSJ International Conference on*. [S.l.: s.n.], 2006. p. 3881–3887.

FREDA, L.; ORIOLO, G. Frontier-based probabilistic strategies for sensor-based exploration. In: *Robotics and Automation. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. [S.l.: s.n.], 2005. p. 3881–3887.

GIRALDI, R. Robôs detectam elevados níveis de radiação em dois reatores da usina de fukushima no japão. *Jornal do Brasil*, 2011. Disponível em: <http://www.jb.com.br/terremoto-no-japao/noticias/2011/04/18/robos-detectam-elevados-niveis-de-radiacao-em-dois-reatores-da-usina-de-fukushima-no-japao/>. Acesso em: agosto 2011.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, v. 4, n. 2, p. 100–107, 1968.

INTUITIVE SURGICAL. *The da Vinci Surgical System*. 2001. Disponível em: <http://www.intuitivesurgical.com/products/davinci_surgical_system/>. Acesso em: novembro 2011.

IROBOT. *iRobot*. 2002. Disponível em: <http://www.irobot.com/>. Acesso em: novembro 2011.

LAVALLE, S. M.; KUFFNER, J. J. Randomized kinodynamic planning. In: *Robotics and Automation. ICRA 1999. Proceedings of the 1999 IEEE International Conference on*. [S.l.: s.n.], 1999. v. 1, p. 473–479.

LAVALLE, S. M.; KUFFNER, J. J. Algorithmic and computational robotics: New directions. In: ____. [S.l.: s.n.], 2001. cap. Rapidly-Exploring Random Trees: Progress and Prospects, p. 293–308.

LOZANO-PÉREZ, T.; WESLEY, M. A. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, v. 22, n. 10, 1979.

MAKARENKO, A. A. et al. An experiment in integrated exploration. In: *Intelligent Robots and Systems. Proceedings of the 2002 IEEE/RSJ International Conference on*. [S.l.: s.n.], 2002. p. 534–539.

NASA. *Mars Exploration Rovers*. 2003. Disponível em: <http://marsrovers.nasa.gov/overview/>. Acesso em: agosto 2011.

ORIOLO, G. et al. The srt method: randomized strategies for exploration. In: *Robotics and Automation. ICRA 2004. Proceedings of the 2004 IEEE International Conference on*. [S.l.: s.n.], 2004. v. 5, p. 4688–4694.

PRESTES, E. *Navegação exploratória baseada em problemas de valores de contorno*. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, 2003.

PRESTES, E.; ENGEL, P. M. Exploration driven by local potential distortions. In: *Intelligent Robots and Systems. Proceedings of the 2011 IEEE/RSJ International Conference on*. [S.l.: s.n.], 2011. p. 1122 –1127.

ROBOMOW. *Robomow*. 1998. Disponível em: <http://www.robomow.com/>. Acesso em: novembro 2011.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. *Introduction to Autonomous Mobile Robots*. Second. [S.l.]: The MIT Press, 2011.

SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. *Robot Modeling and Control*. [S.l.]: John Wiley & Sons, 2006.

STENTZ, A. Optimal and efficient path planning for partially-known environments. In: *Robotics and Automation. ICRA 1994. Proceedings of the 1994 IEEE International Conference on*. [S.l.: s.n.], 1994. v. 4, p. 3310–3317.

THRUN, S. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, v. 99, n. 1, p. 21–71, 1998.

WORLD ROBOTICS. *Executive Summary of World Robotics 2011*. 2011. Disponível em: <http://www.worldrobotics.org/uploads/media/2011_Executive_Summary.pdf>. Acesso em: outubro 2011.

WOWWEE. *Robotics by WowWee*. 2004. Disponível em: <http://www.wowwee.com/en/products/toys/robots/robotics >. Acesso: novembro 2011.

YAMAUCHI, B. A frontier-based approach for autonomous exploration. In: *Computational Intelligence in Robotics and Automation. CIRA 1997. Proceedings of the 1997 IEEE International Symposium on*. [S.l.: s.n.], 1997. p. 146–151.

ANEXO A - TRABALHO DE GRADUAÇÃO 1

Comparação experimental de métodos de exploração de ambientes desconhecidos para robótica móvel

Arthur Vicente Ribacki, Edson Prestes e Silva Júnior

Instituto de Informática – Universidade Federal do Rio Grande do Sul (URGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{avribacki, prestes}@inf.ufrgs.br

Abstract: *Methods for exploring unknown environments have received increasing attention in the academic community and various algorithms are proposed and improved in an impressive speed. However, most of the solutions are validated only through simulations and compared with simplistic approaches. Even when experiments are performed, the use of different platforms makes it difficult to compare and analyze the results. The aim of this paper is to present representative methods that can be implemented in a real robot and compared in terms of runtime, path length and quality of the generated map.*

Resumo: *Métodos de exploração de ambientes desconhecidos têm recebido cada vez mais atenção da comunidade acadêmica e diversos algoritmos são propostos e aperfeiçoados em uma velocidade impressionante. No entanto, a maior parte das soluções é validada apenas através de simulações e comparada com abordagens simplistas. Mesmo quando experimentos são realizados, o uso de diferentes plataformas dificulta a comparação e análise dos resultados. O objetivo deste trabalho é apresentar métodos representativos que possam ser implementados em um robô real e comparados em termos de tempo de execução, caminho percorrido e qualidade do mapa gerado.*

1. Introdução

Antes mesmo de a palavra robô ter sido cunhada, o homem sonhava com a construção de um servo autônomo, incansável e obediente. Na antiguidade, lendas e mitos descrevem criaturas animadas feitas de metal e argila capazes de feitos extraordinários. Diversos autômatos foram construídos e projetados ao longo da história e os avanços tecnológicos do século XX e XXI nos deixam cada vez mais próximos da literatura fantástica de Isaac Asimov. Mesmo se ainda não existam robôs humanoides completamente autônomos circulando entre nós, os robôs já desempenham um papel fundamental em nossa sociedade, que abriga mais de um bilhão de robôs industriais operacionais (World Robotics 2010).

A Organização Internacional para Padronização define robô na ISO 8373 como: *um manipulador automaticamente controlado, reprogramável, multifuncional programável em três ou mais eixos, o qual pode ser fixo no lugar ou móvel para o uso em aplicações de automação industrial*. Essas tarefas são geralmente perigosas, dolorosas, repetitivas ou impossíveis para o ser humano. Exemplos dessas tarefas incluem a exploração de Marte (NASA, 2003) e medidas de níveis de radiação como no desastre ocorrido na usina de Fukushima (GIRALDI, 2011).

Na década de 60 surgiram os primeiros robôs industriais na forma de braços robóticos, como o robô Unimate instalado em 1961 em uma fábrica da General Motors, e AGVs (Veículos Guiados Automaticamente), utilizados para o transporte de carga. Naquela época, o conceito de robô era menos ligado ao conceito de inteligência artificial e mais ao conceito de dispositivo reprogramá-

vel. Seguindo os avanços da microeletrônica e informática, os robôs hoje possuem maior poder de processamento e *autonomia*.

Um robô autônomo deve ser capaz de interagir com o ambiente, percebendo-o através de seus sensores e modificando-o com seus atuadores. As ações tomadas são resultado do *processamento* dos sensores, dando ao robô uma noção de raciocínio, associando-lhe uma inteligência artificial.

Seguindo um importante ramo de pesquisa da robótica atual, este trabalho se situa no campo da robótica móvel autônoma, mais especificamente na área de exploração de ambientes desconhecidos. Seu objetivo é apresentar diferentes estratégias de exploração para que, durante o Trabalho de Graduação 2 (TG2), uma análise comparativa seja realizada por meio de simulações e experimentos em um robô real.

O trabalho é dividido da seguinte maneira. A Seção 2 apresenta uma visão geral da área da robótica móvel. A Seção 3 apresenta as estratégias que serão comparadas no TG2 e a Seção 4 apresenta a metodologia que será utilizada para os experimentos. A Seção 5 discrimina o cronograma a ser seguido durante o TG2.

2. Visão geral

Para que um robô seja plenamente autônomo, é importante que sua atuação independa do ambiente em que ele está inserido. Para isso, ele deve ser capaz de determinar sua posição no mundo (**localização**), ser capaz de construir uma representação interna do ambiente já conhecido (**mapeamento**) e ser capaz de determinar caminhos que o levem a posições desejadas (**movimentação**). Os algoritmos atuais integram estas tarefas de diferentes maneiras (MAKARENKO, WILLIAMS, *et al.*, 2002), como sugere a Figura 1. Nas subseções que seguem, são introduzidos os conceitos de mapeamento, localização, movimentação, SLAM e localização ativa; conceitos de exploração clássica e integrada são detalhados na Seção 3.

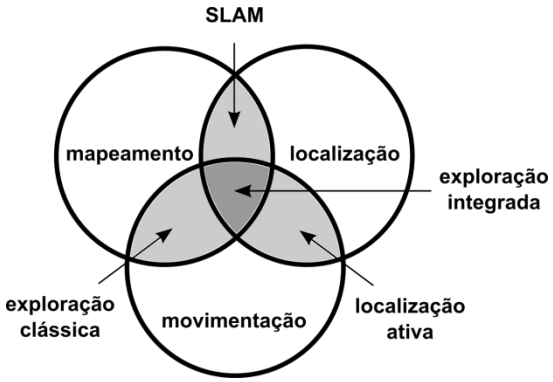


Figura 1 - Tarefas envolvidas no processo de exploração. Figura extraída de (MAKARENKO, WILLIAMS, *et al.*, 2002).

2.1 Mapeamento

A tarefa de mapeamento consiste em guardar em uma representação interna um modelo do mundo utilizando como entrada o retorno dos sensores. Podemos definir duas categorias de modelos: **métricos** e **topológicos**.

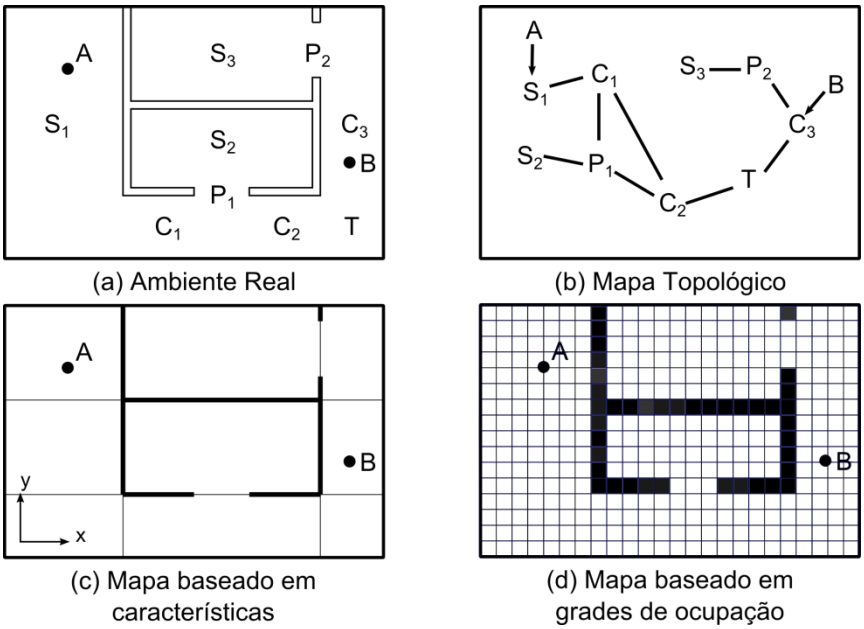


Figura 2 - Um mesmo ambiente (a) representado por um mapa topológico (b), um mapa de características (c) e um mapa baseado em grades de ocupação (d). Figura baseada em (FILLIAT e MEYER, 2003).

Mapas **topológicos** descrevem o ambiente de forma mais abstrata, identificando entidades, suas características e correlações. O mundo da Figura 2(a) é representado por um grafo, como visto na Figura 2(b), onde os nodos são corredores, portas ou salas e as arestas contêm informações que permitem o deslocamento entre os diferentes locais. Como vantagens, podemos citar: (1) sua complexidade depende da complexidade do ambiente, (2) permite um planejamento eficiente e (3) não necessita de posições precisas do robô. No entanto, estes mapas são difíceis de construir e manter, além de não serem adequados a tarefas que exijam uma posição precisa.

Em mapas métricos, o ambiente é representado por um conjunto de objetos com coordenadas cartesianas. Seguindo esta ideia, podemos descrever o ambiente através de **características** obtidas pelos sensores, a fim de representar a forma e posição dos obstáculos. Essas características podem ser pontos específicos, bem como linhas que representem o contorno dos obstáculos. Assim, o mundo da Figura 2(a) pode ser representado por retas como visto na Figura 2(c). Uma dificuldade é extrair as características desejadas de sensores ruidosos, nem sempre confiáveis, e transformá-la em modelos métricos que possam ser utilizados.

Outra possibilidade é representar o espaço acessível ao robô. A técnica mais difundida é a **grade de ocupação** (ELFES, 1989), vista na Figura 2(d), onde cada célula de uma grade regular representa uma porção do ambiente visto na Figura 2(a). Cada célula guarda uma probabilidade de estar ocupada, a qual é utilizada para classificar a região como: livre, ocupada ou desconhecida. A grande vantagem deste tipo de mapa é a facilidade de manipulação e a rápida atualização utilizando diferentes sensores, enquanto que os principais problemas são granularidade, escalabilidade e extensibilidade. Existem diversas técnicas para atualização e representação da probabilidade de ocupação de cada célula como certeza fuzzy, bayes, heurística, gaussiana ou frequência.

As vantagens e desvantagens destes dois tipos de mapas são ortogonais (THRUN, 1997). Para o processo de exploração, a facilidade de construção e manutenção desempenha um papel impor-

tante; sendo assim, uma parte considerável dos algoritmos de exploração opta por mapas métricos, mais especificamente pelas grades de ocupação.

2.2 Localização

Para que um robô atinja seus objetivos, é fundamental saber sua localização no ambiente, pois para saber como chegar a uma determinada posição ele deve primeiro saber onde se encontra. Para isso, ele deve utilizar as informações oriundas de seus sensores *proprioceptivos* ou *extraceptivos*. Sensores proprioceptivos fornecem informações internas ao sistema, e.g. temperatura, velocidade, aceleração, número de rotações das rodas; enquanto extraceptivos fornecem informações do ambiente, e.g. distância dos obstáculos, intensidade luminosa, amplitude sonora.

A localização pode ser classificada como local ou global. A *localização local* consiste em estimar uma nova posição a partir de uma posição conhecida[1]. O uso apenas de técnicas de odometria leva ao método conhecido como *dead-reckoning*, que consiste em integrar o número de rotações das rodas a fim de estimar o deslocamento relativo do robô e consequentemente sua posição atual. No entanto, por envolver uma integração, esta técnica está sujeita ao acúmulo de erros sistemáticos e não sistemáticos. Erros sistemáticos, como desalinhamento ou diferença de tamanho das rodas, são determinísticos e podem ser corrigidos por uma correta calibração do sistema. Erros não sistemáticos, como derrapagem, não podem ser previstos e adicionam incertezas na localização. Assim, a utilização exclusiva de dados proprioceptivos funciona apenas em curto prazo e a estimação precisa ser corrigida regularmente através de informações do ambiente. A Figura 3(a) mostra um exemplo de um mapa gerado apenas com uso de odometria, enquanto no mapa da Figura 3(b) é utilizada uma técnica de localização. Note como o mapa sofre deformações devido ao erro na estimação da localização do robô.

Figura 3 – Um ambiente utilizando apenas técnicas de odometria para se localizar (a) e utilizando técnicas de localização para estimar sua posição (b). Figura extraída de (THRUN, 1997).

As técnicas de localização podem ser baseadas em uma única hipótese ou em múltiplas hipóteses. Técnicas baseadas em uma única hipótese estimam uma única posição a cada iteração. Isto facilita o processo de tomada de decisões, pois não há ambiguidade quanto à localização do robô. No entanto, se o objetivo é *localização global*, faz-se necessário o uso de múltiplas hipóteses.

[1] Para o processo de construção de mapas métricos costuma-se associar a posição inicial ao ponto (0,0) do plano cartesiano. Desse modo, o robô parte de uma posição conhecida ao iniciar a exploração.

Na localização global, o robô não tem conhecimento prévio de sua posição. A partir das leituras de seus sensores e em posse de um mapa do ambiente, ele determina neste mapa a posição mais provável de produzir uma resposta sensorial igual à observada. A necessidade de múltiplas hipóteses vem do problema conhecido como *perception aliasing*, i.e., para um dado sistema sensorial, diferentes posições do mapa podem gerar respostas muito semelhantes, fazendo com que dois lugares distintos pareçam iguais. Assim, várias posições são rastreadas, cada uma com uma probabilidade associada. Conforme o robô avança e recebe mais informações, algumas posições possuem sua probabilidade aumentada e outras diminuídas, reduzindo a ambiguidade da localização. Quando o robô deve tomar uma decisão com base na sua localização, a técnica mais simples consiste em utilizar a posição com a maior probabilidade associada.

2.3 Movimentação

A movimentação consiste na determinação de caminhos e trajetórias entre uma postura inicial q_{inicio} e uma postura final q_{fim}. Para um ambiente planar ($\mathbb{R}^2$), a **postura** de um robô móvel pode ser definida pela tupla (x, y, θ), i.e., sua posição e orientação. Considerando Q o conjunto de todas as posturas possíveis e Q_{livre} o conjunto de todas as posturas nas quais o robô não colide com obstáculos, a solução para um problema de *planejamento de caminhos* é uma função contínua $c(s)$ parametrizada entre zero e um, tal que:

$$c(s) : [0,1] \rightarrow Q, onde\ c(0) = q_{inicio}, c(1) = q_{fim}\ e\ c(s)\ \epsilon\ Q_{livre}\ \forall s\ \in [0,1]$$

Isto quer dizer que a solução é uma sequência de posturas que o robô deve assumir de modo que ele saia de q_{inicio} e chegue a q_{fim} sem colidir com obstáculos. Esta função não informa *como* realizar o percurso, e sim apenas a *forma* que o percurso deve possuir.

Se o caminho for parametrizado em função do tempo t, então $c(t)$ é uma *trajetória*, e a velocidade e aceleração podem ser calculadas pela primeira e segunda derivada em relação ao tempo. Assim, além de descrever a forma, uma trajetória descreve *como* o percurso deve ser realizado.

Uma das técnicas mais simples de planejamento de caminho consiste em seguir em linha reta em direção à posição objetivo, contornando obstáculos caso estes sejam encontrados. Este é o princípio dos algoritmos conhecidos como Bug que, apesar de simples, sempre encontram uma solução caso esta exista (CHOSET, LYNCH, *et al.*, 2005).

Outro conjunto de técnicas tem como base o uso de funções de potenciais. Para melhor compreensão, faz-se analogia a potenciais elétricos. Nesta analogia, o robô é visto como uma partícula móvel carregada positivamente, em um ambiente onde obstáculos são cargas positivas fixas, exercendo forças de repulsão, e o objetivo é uma carga negativa fixa, a qual exerce uma força de atração. Calculando o gradiente do potencial definido pelo ambiente, obtém-se um campo vetorial que define o caminho até o objetivo a partir de qualquer posição.

2.4 Localização e Mapeamento Simultâneo

As técnicas de localização e mapeamento simultâneo, conhecidas por SLAM (*Simultaneous Localization and Mapping*), têm o seguinte objetivo: partindo de uma posição desconhecida, em um ambiente desconhecido, o robô deve ser capaz de construir um mapa, utilizando apenas observações relativas do ambiente e utilizar este mapa incompleto para se localizar simultaneamente.

A dificuldade do problema vem do fato de um mapa ser necessário para estimar a posição do robô ao mesmo tempo em que a posição é necessária para construção do mapa.

Alguns algoritmos de SLAM precisam ser executados após a completa aquisição dos dados, devido à sua alta complexidade. Essa classe de algoritmos é dita *off-line*, enquanto aqueles capazes de executarem durante a aquisição dos dados são chamados *online.* As técnicas mais difundidas para solucionar o problema de SLAM são baseadas no uso do Filtro de Kalman e do Filtro de Partículas.

2.5 Localização Ativa

A *localização ativa* assume que a rotina de localização tem acesso total ou parcial sobre os atuadores do robô, provendo a oportunidade de aumentar a eficiência e robustez da localização (BUGARD, FOX e THRUN, 1997). O objetivo é determinar onde o robô deve ir e em que direção ele deve orientar seus sensores de forma a melhor localizar sua posição.

Um trabalho importante nesta área foi desenvolvido por Bugard et al. (BUGARD, FOX e THRUN, 1997) que definiram um método de localização ativa com base na localização de Markov, um método passivo. Na localização de Markov, a posição do robô é representada através de uma função densidade de probabilidade arbitrária, atualizada a partir dos dados sensoriais. Na localização ativa, as posições e orientações que o robô deve atingir são determinadas de modo a diminuir a entropia associada à função densidade de probabilidade.

3. Exploração de ambientes desconhecidos

Integrando-se as tarefas de movimentação e mapeamento obtém-se um algoritmo de **exploração clássica**, definida como (YAMAUCHI, 1997):

Exploração é o ato de se mover através de um ambiente desconhecido construindo um mapa que pode ser usado para navegação subsequente. Uma boa estratégia de exploração é aquela que gera um mapa completo ou quase completo em um tempo razoável.

Sendo assim, a ideia principal é minimizar o trajeto percorrido enquanto se maximiza a informação obtida. Note que este conceito não considera a *qualidade* do mapa. Sem considerarmos a certeza da localização, o mapa gerado pode ser inútil uma vez que erros de odometria vão sendo acumulados conforme o processo de exploração avança. Isto é exemplificado na Figura 3, onde o mapa (a) é obtido utilizando apenas técnicas de odometria e o mapa (b) utilizando uma técnica de SLAM.

Assim, além de maximizar a quantidade de informação obtida, uma boa estratégia de exploração deve considerar a *qualidade da localização* para determinar a trajetória a ser efetuada. Este tipo de estratégia é chamado de **exploração integrada** e tem como objetivo orientar o robô em um ambiente desconhecido, gerando um percurso que maximize a certeza de sua localização e permita um mapeamento completo e correto. Note que o comprimento do caminho não precisa ser o mínimo. Por exemplo, pode ser necessário retornar a lugares já explorados para aumentar a certeza da localização.

Uma quantidade representativa de estratégias utiliza métodos de *planejamento de caminhos* para orientar o robô em direção a regiões desconhecidas do ambiente. Utilizando grades de ocupação, Yamauchi (YAMAUCHI, 1997) definiu uma *fronteira* como o limite entre a região explorada livre de obstáculos e a região não explorada, conforme visto na Figura 4.

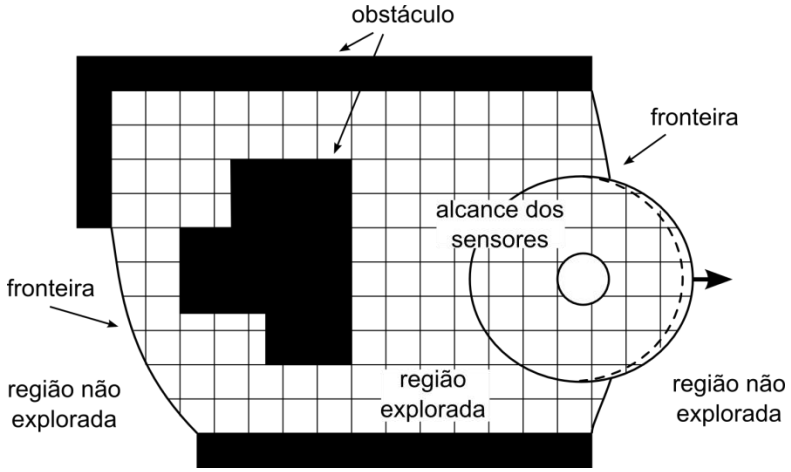


Figura 4 – As células pertencentes à fronteira são as células livres próximas de células desconhecidas. Figura extraída de (PRESTES, 2003).

Através de uma função de custo-benefício, define-se qual a fronteira que o robô deve atingir. O custo pode ser visto como a distância até a fronteira e o benefício pode levar em conta fatores como a expectativa de aumento do mapa, a observação de regiões específicas como portas ou corredores, ou até mesmo o aumento na certeza da localização.

A seguir, descreveremos três algoritmos de exploração. O primeiro utiliza soluções de Problemas de Valor de Contorno (PVC) para guiar o robô até as fronteiras (PRESTES, 2003). O segundo é baseado no método SRT (*Sensor-based Random Tree*), que utiliza a geração aleatória e incremental de uma estrutura de dados em forma de árvore (ORIOLO, VENDITTELI, *et al.*, 2004). O terceiro é um método de exploração integrada que utiliza duas camadas: uma deliberativa e uma reativa (JULIÁ, ÓSCAR, *et al.*, 2010).

3.1 Exploração utilizando soluções de PVC

O método de exploração de ambientes baseado em funções harmônicas é uma extensão do método de planejamento de caminhos baseado em funções harmônicas de Connoly e Grupen (CONNOLY e GRUPEN, 1993).

Para o planejamento, é calculada a solução numérica da equação de Laplace $\nabla^2 p(\boldsymbol{r}) = 0$ com condições de contorno de Dirichlet, onde $p(\boldsymbol{r})$ é o potencial na posição $\boldsymbol{r} \in \mathbb{R}^2$. O ambiente é dividido em uma malha regular (*grade de ocupação*), onde cada célula representa uma porção quadrada do ambiente e armazena um valor de potencial. Células ocupadas (obstáculos) têm seu potencial definido em **1** (alto), enquanto células objetivos têm seu potencial definido em **0** (baixo), caracterizando as condições de contorno de Dirichlet. As células restantes têm seu potencial calculado através de iterações sucessivas utilizando-se um método de diferenças finitas. Uma característica importante é que o potencial gerado é livre de mínimos locais e os caminhos produzidos são suaves.

Este método de planejamento é estendido de forma a guiar o robô para as fronteiras do ambiente (PRESTES, 2003). Isto faz com que ao invés de definir uma única posição que o robô deve atingir, são definidas várias posições que precisam ser visitadas. Além disso, esta extensão utiliza um conjunto mais genérico de funções, também geradoras de campos potenciais livres de mínimos locais, descrita por $\nabla^2 p + F(\nabla \mathrm{p}) = 0$, onde $F(\boldsymbol{v})$ é qualquer função contínua com $F(\mathbf{0}) = 0$. Como exemplo, a adição do termo $F(\nabla \mathrm{p}) = \nabla p \cdot \boldsymbol{v}$ aumenta a desempenho do algoritmo no caso de ambientes esparsos[2], pois este define uma direção preferencial definida pelo vetor constante $\boldsymbol{v}$ e que independe da presença de obstáculos.

Esta extensão leva ao algoritmo proposto por Prestes et al.:

1. **enquanto** *não explorou todo ambiente* **faça**
2. *ativa e obtém a leitura dos sensores*
3. *atualiza a probabilidade de ocupação das células dentro do campo de visão do robô*
4. *atualiza o potencial de cada célula*
5. *calcula o vetor gradiente descendente a partir da sua posição corrente no mapa*
6. *desloca-se seguindo a direção definida por este gradiente.*

Linha 1: A condição de parada consiste em verificar se não existe alguma célula *acessível* e livre ao lado de uma célula não explorada. Para verificar se uma célula livre é acessível ou não, foi utilizada uma variação do algoritmo de crescimento de regiões, que segmenta o espaço livre em uma região conectada (acessível) e outra desconecta (inacessível).

Linha 2: Utilizando o retorno dos sensores, define-se um cone de atualização com base na distância medida, no ângulo de abertura e postura corrente do robô.

Linha 3: As células dentro do cone têm seu atributo de certeza reduzido em 1, enquanto as células no limite do cone têm seu atributo de certeza incrementado de 3. Quando este atributo for maior que dois, a célula é considerada ocupada.

Linha 4: Como no planejamento de caminhos, os obstáculos são definidos com potencial alto (1), enquanto que regiões desconhecidas (o objetivo) com potencial baixo (0). Para o cálculo do potencial das demais células, utiliza-se o método Gauss-Seidel para resolução numérica da equação de Laplace, o que resulta na equação (1), onde $p_{i,j}^{t}$ é o potencial da célula de posição (i, j) na iteração t. Para o processo de exploração não é necessário que o potencial tenha convergido e um número fixo de iterações é utilizado a cada passo.

$$p_{i,j}^{t+1} = \frac{1}{4}\left(p_{i-1,j}^{t+1} + p_{i+1,j}^{t} + p_{i,j-1}^{t+1} + p_{i,j+1}^{t}\right) \quad (1)$$

Linha 5: O vetor gradiente na posição (i, j) é calculado com a equação (2).

$$\boldsymbol{v} = \left(v_x, v_y\right) = \left(\frac{p_{i-1,j} - p_{i+1,j}}{2}, \frac{p_{i,j-1} - p_{i,j+1}}{2}\right) \quad (2)$$

Linha 6: A direção definida pelo gradiente é calculada com a equação (3). A velocidade é uma função da proximidade dos obstáculos, da diferença angular entre a direção desejada e a direção atual e da velocidade definida no passo anterior.

[2] Um ambiente *denso* é definido como aquele em que na maior parte do tempo existe algum obstáculo no campo de visão dos sensores do robô, ao contrário de ambientes *esparsos.*

$$\phi = arctan\left(\frac{v_y}{v_x}\right) \quad (3)$$

3.2 Exploração utilizando Sensor-based Random Tree (SRT)

O método SRT (ORIOLO, VENDITTELI, *et al.*, 2004) se baseia na construção incremental e aleatória de uma árvore que representa o roteiro da exploração com uma região segura associada. A *região segura* é o equivalente do conceito de região livre definido pela grade de ocupação, enquanto o resto, obstáculos e regiões não exploradas, pertencem a uma *região não segura*. Este algoritmo se diferencia por não fazer distinção entre obstáculos e regiões não exploradas, não empregando o conceito de fronteira. Freda et al. apresentam uma otimização que leva em conta as fronteiras para influenciar as direções aleatórias geradas (FREDA e ORIOLO, 2005).

Este método é probabilístico e inspirado em algoritmos como o *Rapidly-exploring Random Trees* (RRT) (LA VALLE e KUFFNER, 2001). Ele tem como vantagem a simplicidade e o fato de que qualquer sequência de ações será executada eventualmente, implicando que se alguma solução existir, esta será encontrada se o tempo necessário for fornecido.

O método utiliza as seguintes suposições: (I) o ambiente é planar, i.e., $\mathbb{R}^2$ ou um subconjunto conectado de $\mathbb{R}^2$; (II) o robô é um disco livre para se deslocar em qualquer direção (*holonômico*); (III) o robô sempre sabe sua postura q; (IV) para cada q, o sistema sensorial é capaz de determinar o espaço livre ao redor do robô, definindo uma região segura local $\boldsymbol{\mathcal{S}}(q)$, a qual deve possuir um formato de estrela.

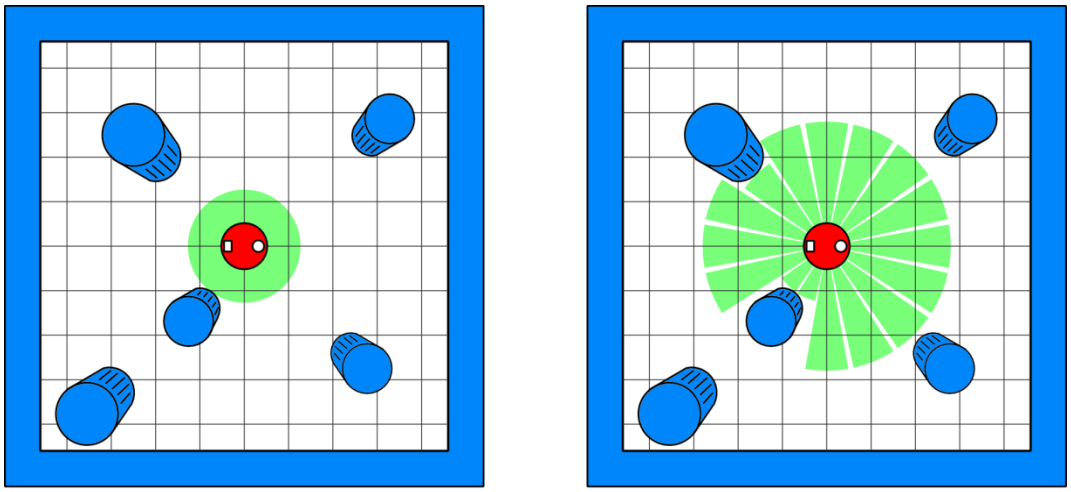
O algoritmo geral pode ser descrito como:

1. **SRT(** $q_{inicial}$**,** K_{max}**,** I_{max}**,** α**,** d_{min} **)**
2. $q_{atual} = q_{inicial}$
3. **para** k = 1 **até** K_{max}
4. $\boldsymbol{\mathcal{S}}(q_{atual}) =$ *PERCEPÇÃO(* q_{atual} *)*
5. *ADICIONA*($\boldsymbol{\mathcal{T}}$,(q_{atual}, $\boldsymbol{\mathcal{S}}(q_{atual})$))
6. i = 0
7. **repetir**
8. θ_{rand} = *DIREÇÃO_ALEATÓRIA*
9. r = *RAIO*($\boldsymbol{\mathcal{S}}(q_{atual})$,$\theta_{rand}$)
10. $q_{candidato}$ = *DESLOCAR*(q_{atual}, θ_{rand}, α*r)
11. i = i + 1
12. **até** *VALIDAR*($q_{candidato}$, d_{min}, $\boldsymbol{\mathcal{T}}$) **ou** i = I_{max}
13. **se** *VALIDAR*($q_{candidato}$, d_{min}, $\boldsymbol{\mathcal{T}}$) **então**
14. *MOVER_PARA*($q_{candidato}$)
15. $q_{atual} = q_{candidato}$
16. **senão**
17. *MOVER_PARA*(q_{atual}.pai)
18. $q_{atual} = q_{atual}$.pai
19. **retorna** $\boldsymbol{\mathcal{T}}$

A cada iteração k o robô percebe seu ambiente (linha 4) e definir uma Região Segura Local (RSL) $\boldsymbol{\mathcal{S}}$. Após, adiciona na árvore $\boldsymbol{\mathcal{T}}$ um novo nodo, formado pelo par RSL e postura, como filho do

nodo corrente (linha 5). No loop das linhas 7 a 12, uma direção aleatória θ_{rand} é escolhida. A função *RAIO* retorna a distância até a borda de $\mathcal{S}$ na direção θ_{rand}, enquanto que o fator α é utilizado para que $q_{candidato}$ sempre caia dentro da RSL. Se esta for válida, o robô avança para ela (linha 14), caso contrário, uma nova é escolhida, até I_{max} tentativas. Se nenhuma posição válida foi encontrada, o robô volta para a posição anterior na árvore (linha 17). Para uma posição candidata ser válida, esta deve: (1) estar a uma distância d_{min} da posição atual e (2) não estar contida em nenhuma RSL de um nodo de $\mathcal{T}$. Note que a imposição (1) faz com que o robô se afaste naturalmente dos obstáculos, e a imposição (2) faz com que o robô retorne ao explorar completamente o espaço livre. Ao terminar a exploração, o robô se encontrará na posição $q_{inicial}$, caracterizando um mecanismo automático de *regresso ao inicio*. O mapa resultante é a união de todas RSL, as quais definem o espaço acessível ao robô.

Este algoritmo geral depende fortemente da função *PERCEPÇÃO* utilizada, a qual deve gerar uma região em forma de estrela e depende do hardware disponível no robô. Uma técnica *conservativa* consiste em definir a RSL como um disco com raio igual à menor distância obtida pelos sensores. Chamada SRT-Bola, esta versão é mais apropriada para sensores ruidosos. Uma versão mais *confiante*, chamada SRT-Estrela, consiste em definir a RSL como um conjunto de cones centrados no robô, dando origem a uma região com formato de estrela. Cada cone possui como raio a distância mínima dos sensores participantes do cone. Esta versão apresenta em média uma distância percorrida menor, menor número de nodos e maior fidelidade ao reproduzir o espaço livre, sendo apropriada apenas quando o retorno dos sensores for preciso. A Figura 5 mostra a RSL para um mesmo ambiente utilizando os dois métodos de percepção.



(a) Região Segura Local para SRT-Bola (b) Região Segura Local para SRT-Estrela

Figura 5 – Diferença entre as RSL segundo a percepção SRT-Bola (a) e SRT-Estrela (b). Figura extraída de (ORIOLO, VENDITTELI, *et al.*, 2004).

3.3 Exploração integrada através do método híbrido

Juliá et al. propuseram uma solução para o problema da exploração integrada utilizando múltiplos robôs (JULIÁ, ÓSCAR, *et al.*, 2010). Esta solução faz uso de um módulo de SLAM, de uma camada dita *deliberativa* e uma camada dita *reativa*, resultando no que eles chamaram de *método híbrido*. Neste trabalho uma versão simplificada é introduzida considerando apenas um robô.

O módulo de *SLAM* é responsável pela construção do mapa e localização do robô no mesmo. O método utilizado é o FastSLAM, que utiliza técnicas de visão para reconhecer *landmarks* e um filtro de partículas. Como o mapa de landmarks não representa o espaço livre, um mapa auxiliar do tipo grade de ocupação é utilizado, permitindo a identificação de fronteiras. Este módulo também deve registrar as células onde o robô estava bem localizado, para que quando a certeza estiver muito baixa ele retorne para estas posições. Toda vez que o erro de localização estiver abaixo de certo limiar, a célula onde o robô está situado é marcada como *posição precisa*.

A camada *reativa* faz uso de técnicas comportamentais para realizar a exploração em um nível local. Os comportamentos básicos são: (1) ir para regiões não exploradas, (2) ir para fronteiras, (3) evitar obstáculos, (4) ir para porta e (5) ir para posições precisas.

As células consideradas pela camada reativa são apenas aquelas dentro da *Região Segura Esperada* (RSE). Esta região é definida como o conjunto de células, livres ou desconhecidas, que podem ser unidas com a posição do robô através de uma linha reta sem intersectar algum objeto e até uma distância máxima (d_{rse}). Assim, a RSE pode ser vista como resultado do uso de um conjunto de sensores virtuais, com uma abrangência de 360° e alcance d_{rse}. Esta distância independe do alcance dos sensores reais d_s, estes podendo ter uma abrangência angular limitada. Este alcance maior faz com que a RSE contenha células não exploradas, ao contrário da Região Segura Local utilizada pelo algoritmo SRT, a qual está relacionada com os sensores reais. Por não considerar as células atrás de obstáculos ou passagens muito estreitas, as forças resultantes não estão sujeitas a mínimos locais.

Outro conceito importante é o de *Células Portas*, que são as células livres pertencentes à RSE vizinhas a células livres não pertencentes à RSE. Esses conceitos podem ser vistos na Figura 6.

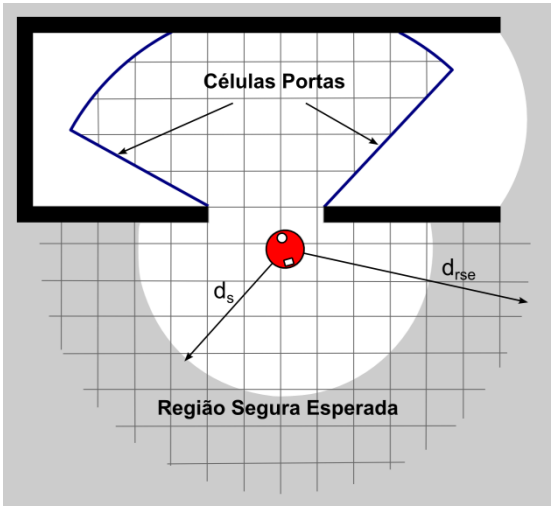


Figura 6 - Conceitos de Região Segura Esperada e Células Portas. Figura extraída de (JULIÁ, ÓSCAR, *et al.*, 2010).

A camada *deliberativa* executa em paralelo e é responsável por ativar e desativar os comportamentos básicos da camada reativa, realizando o chaveamento entre três estados: explorar a RSE atual (comportamentos 1, 2 e 3), trocar de região (3 e 4) e localização ativa (3 e 5). Esta decisão é baseada na certeza da localização e na análise de uma *árvore de exploração*. A cada iteração da camada deliberativa, de frequência menor que a reativa, uma nova árvore é criada.

Nesta árvore, cada nodo é associado a uma posição espacial, sendo que a raiz se encontra na posição atual do robô. Podem ser adicionados dois tipos de nodos: ramos e folhas. Para cada *porta* encontrada, um ramo é adicionado. As folhas representam o objetivo: mapeamento ou

localização ativa. Se o robô está bem localizado, para cada *fronteira* uma folha é adicionada. Caso contrário, cada grupo de *células bem localizadas* gera uma folha.

A Figura 7 apresenta um exemplo de construção da árvore de exploração considerando que o robô está bem localizado. A partir da posição do robô (nodo raiz), determina-se a RSE correspondente. Para cada porta (células portas agrupadas por proximidade) ramos são adicionados. Para cada fronteira, uma folha é adicionada. Seleciona-se a porta mais próxima e calcula-se a RSE excluindo as células já pertencentes a outras RSE. Portas (ramos) e fronteiras (folhas) são adicionadas e o processo se repete até que não haja mais portas.

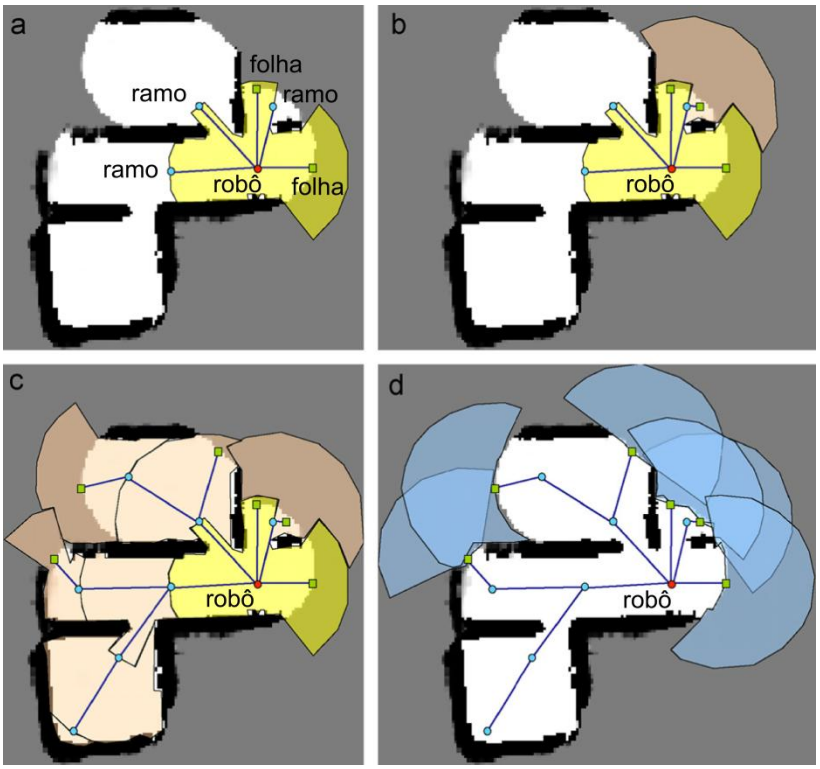


Figura 7 – Exemplo de árvore de exploração. (a) O nodo raiz é adicionado, bem como as portas e fronteiras dentro da RSE. (b) A partir do ramo mais próximo, a RSE é calculada e filhos adicionados. (c) Processo continua até não ter mais portas. (d) Para cada folha, calcula-se o número esperado de células não exploradas. Figura extraída de (JULIÁ, ÓSCAR, *et al.*, 2010).

Após, associa-se a cada folha um peso proporcional ao número de células de interesse e inversamente proporcional ao quadrado do custo para atingi-la. As células de interesse são o número de células não exploradas (mapeamento) ou a quantidade de células bem localizadas (localização ativa) dentro da RSE vista de cada folha. Os valores são propagados para os ramos, escolhendo-se o valor máximo entre os nodos filhos. Finalmente, o estado que o robô deve assumir é decidido com base nos pesos dos nodos conectados diretamente à raiz, os quais podem ser folhas ou ramos, e da qualidade da localização:

- *Explorar a RSE atual*: quando a **localização é boa** e o nodo de maior valor conectado à raiz é uma **folha**.
- *Trocar de região*: quando o nodo de maior valor conectado à raiz é um **ramo**. Seleciona-se a porta para o comportamento *ir para porta* com base neste ramo. Este estado é ativado **independentemente do nível de localização**, pois a árvore já é construída com base na localização.
- *Localização ativa:* quando a **localização é ruim** e o nodo de maior valor conectado à raiz é uma **folha**.

4. Metodologia

Em nosso instituto temos a disposição o robô **Pioneer P3-DX** da empresa Adept Mobile Robots. É um robô com duas rodas diferencial e possui sensores do tipo sonar, laser e visão, além de um módulo de odometria capaz de estimar deslocamentos. Ele é programável através da biblioteca C++ ARIA (*Advanced Robot Interface for Applications*), a qual permite controlar os atuadores e receber informações dos sensores. Além disso, a implementação dos algoritmos será toda em C++ e as interfaces desenvolvidas utilizando o framework Qt da Nokia.

O projeto será realizado através das seguintes etapas:

1. Levantamento bibliográfico.
2. Implementação dos algoritmos:
    a. Método baseado em PVC.
    b. Método SRT.
    c. Método híbrido.
3. Validação dos algoritmos utilizando o simulador MobileSim.
4. Experimentos comparativos entre os algoritmos utilizando o robô real.
5. Análise dos resultados experimentais.
6. Escrita do relatório para o TG2.

Durante a análise experimental, dois conjuntos de testes serão realizados:

1) As soluções apresentadas serão comparadas em termos de *tempo de execução*, *comprimento do caminho percorrido* e *qualidade do mapa.* Para que as comparações sejam justas, o algoritmo híbrido terá seu módulo de SLAM desativado.
2) Ativando-se o módulo SLAM para o terceiro algoritmo, será analisada a melhora em termos de qualidade do mapa gerado e o impacto no tempo de execução e caminho percorrido.

5. Cronograma previsto

Tabela 1 - Cronograma de atividades previsto para implementação do trabalho descrito.

	Meses / Semanas													
	Agosto		**Setembro**				**Outubro**					**Novembro**		
Etapa	**15/8**	**22/8**	**29/8**	**5/9**	**12/9**	**19/9**	**26/9**	**3/10**	**10/10**	**17/10**	**24/10**	**31/10**	**7/11**	**14/11**
1	x	x												
2.a			x	x										
2.b				x	x									
2.c					x	x								
3							x	x						
4									x	x	x			
5											x	x	x	
6												x	x	x

6. Conclusão

A rápida evolução no ramo da robótica móvel nem sempre é acompanhada por testes extensivos ou comparações. Os três algoritmos de exploração apresentados serão comparados, possibilitando a criação de um perfil de seus pontos fortes e fracos. Espera-se que este estudo permita avaliar com mais precisão as vantagens e desvantagens de cada estratégia e o impacto de melhoras propostas em futuros trabalhos.

7. Bibliografia

BUGARD, W.; FOX, D.; THRUN, S. **Active mobile robot localization by entropy minimization**. Advanced Mobile Robots, 1999. Proceedings from the Second EUROMICRO workshop on. [S.l.]: [s.n.]. 1997. p. 155 - 162.

CHOSET, H. et al. **Principles of Robot Motion:** Theory, Algorithms, and Implementation. [S.l.]: The MIT Press, 2005.

CONNOLY, C.; GRUPEN, R. On the application of harmonic functions to robotics. **Journal of Robotics v.10**, p. 931 - 946, 1993.

ELFES, A. Using occupancy grids for mobile robot perception and navigation. **Computer**, p. 46-57, Junho 1989.

ENGELSON, S. P.; MCDERMOTT, D. V. **Error correction in mobile robot map-learning**. Robotics and Automation. ICRA 1992. Proceedings of the 1992 IEEE International Conference on. [S.l.]: [s.n.]. 1992. p. 2555 - 2560.

FILLIAT, D.; MEYER, J.-A. Map-based navigation in mobile robots: I. A review of localization strategies. **Cognitive Systems Research**, n. 4, p. 243 - 282, 2003.

FREDA, L.; ORIOLO, G. **Frontier-based probabilistic strategies for sensor-based exploration**. Robotics and Automation. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. [S.l.]: [s.n.]. 2005. p. 3881 - 3887.

GIRALDI, R. Robôs detectam elevados níveis de radiação em dois reatores da Usina de Fukushima no Japão. **Jornal do Brasil**, 2011. Disponivel em: <http://www.jb.com.br/terremoto-no-japao/noticias/2011/04/18/robos-detectam-elevados-niveis-de-radiacao-em-dois-reatores-da-usina-de-fukushima-no-japao/>. Acesso em: 18 Agosto 2011.

JULIÁ, M. et al. A hybrid solution for the multi-robot integrated exploration problem. **Engineering Applications of Artifical Intelligence**, p. 473-486, Janeiro 2010.

LA VALLE, S. M.; KUFFNER, J. J. Rapidly-Exploring Random Trees: Progress and Prospects. In: LA VALLE, S. M.; KUFFNER, J. J. **Algorithmic and Computational Robotics:** New Directions. [S.l.]: [s.n.], 2001. Cap. 10, p. 293 - 308.

MAKARENKO, A. A. et al. **An experiment in integrated exploration**. Intelligent Robots and Systems. Proceedings of the 2002 IEEE/RSJ International Conference on. [S.l.]: [s.n.]. 2002. p. 534 - 539.

NASA. Mars Exploration Rovers: Overview. **Mars Exploration Rovers**, 2003. Disponivel em: <http://marsrovers.nasa.gov/overview/>. Acesso em: 18 Agosto 2011.

ORIOLO, G. et al. **The SRT method:** randomized strategies for exploration. Robotics and Automation. ICRA 2004. Proceedings of the 2004 IEEE International Conference on. [S.l.]: [s.n.]. 2004. p. 4688 - 4694 Vol. 5.

PRESTES, E. **Navegação exploratória baseada em problemas de valores de contorno**. Porto Alegre: PPGC da UFRGS, 2003.

THRUN, S. Learning maps for indoor mobile robot navigation. **Artificial Intelligence**, 1997.

YAMAUCHI, B. **A frontier-based approach for autonomous exploration**. Computational Intelligence in Robotics and Automation. CIRA 1997. Proceedings of the 1997 IEEE International Symposium on. [S.l.]: [s.n.]. 1997. p. 146-151.