CENTRO UNIVERSITÁRIO DE ARARAQUARA - UNIARA
DEPARTAMENTO DE CIÊNCIAS DA ADMINISTRAÇÃO E TECNOLOGIA

ENGENHARIA DE COMPUTAÇÃO

FLAVIO LUIZ DOS SANTOS DE SOUZA

IMPLEMENTAÇÃO DE UM ALGORITMO DE LOCALIZAÇÃO DE ROBÔS MÓVEIS

ARARAQUARA

2012

FLAVIO LUIZ DOS SANTOS DE SOUZA

IMPLEMENTAÇÃO DE UM ALGORITMO DE LOCALIZAÇÃO DE ROBÔS MÓVEIS

Trabalho de Conclusão de Curso (TCC) apresentado ao Departamento de Ciências da Administração e Tecnologia, do Centro Universitário de Araraquara - UNIARA, como parte dos requisitos para obtenção do título Bacharel em Engenharia de Computação

Orientador: Prof. Dr. Rodrigo Elias Bianchi

ARARAQUARA

2012

TERMO DE AUTENTICIDADE

Eu, FLAVIO LUIZ DOS SANTOS DE SOUZA, RG: 32.303.149-3, aluno regularmente matriculado no Curso de Engenharia de Computação do Centro Universitário de Araraquara - UNIARA, declaro ser o autor do texto apresentado como Trabalho de Conclusão de Curso - TCC com o título "IMPLEMENTAÇÃO DE UM ALGORITMO DE LOCALIZAÇÃO DE ROBÔS MÓVEIS".

Afirmo, também, ter seguido as normas da ABNT referentes às citações textuais que utilizei e das quais eu não sou autor, dessa forma, creditando a autoria a seus verdadeiros autores (Lei n.9.610, 19/02/1998).

Através dessa declaração dou ciência de minha responsabilidade sobre o texto apresentado e assumo qualquer responsabilidade por eventuais problemas legais, no tocante aos direitos autorais e originalidade do texto.

Araraquara, 13 de Novembro de 2012

Assinatura: _____

FOLHA DE APROVAÇÃO

FLAVIO LUIZ DOS SANTOS DE SOUZA

IMPLEMENTAÇÃO DE UM ALGORITMO DE LOCALIZAÇÃO DE ROBÔS MÓVEIS

Trabalho de Conclusão de Curso, apresentado ao Programa de Graduação, para obtenção do título de bacharel em Engenharia de Computação.
Centro Universitário de Araraquara
Araraquara, 13 de Novembro de 2012

Orientador:

Prof. Dr. Rodrigo Elias Bianchi _____

Examinador:

Prof. Msc. Rodrigo Daniel Malara _____

Nota: _____

"Tenho em mim todos os sonhos do mundo".

Fernando Pessoa

Trabalho dedicado a todo sonhador que amanhece focado em sua longa e árdua jornada, em busca de seus valores e objetivos.

AGRADECIMENTOS

A Deus, por tudo que tem me proporciona na vida. Aos meus pais, Pedro A. de Souza e Francisca S. de Souza, para os quais palavras não seriam suficientes para descrever o tanto que aprendi com exemplos de vida, família, realização do possível e impossível, e pelo amor incondicional que sinto por eles. À minha noiva Gleice Kele, pela compreensão, apoio, simplicidade, humildade e por estar próxima em todos os momentos. À minha irmã Patrícia e cunhado Cristiano, por me ajudarem a manter meus pés no chão. Não poderia de deixar de agradecer aos meus queridos padrinhos, José Henrique e Ana Lúcia, e Eraldo e Irene, pelos incentivos, apoios, cobranças e puxões de orelha (certamente deram resultados e acredito que na maioria das vezes foram positivos).

Ao Prof. Coord. Msc. Rodrigo Malara por ter acreditado, apoiado, incentivado e contribuído nas loucuras destes e de outros projetos que me proporcionaram valiosos ensinamentos, devo-lhe muito; também ao meu orientador Prof. Dr. Rodrigo Bianchi por ter assumido comigo o compromisso deste trabalho, pela sua paciência, disponibilidade, companheirismo e amizade; finalmente, à turma de Engenharia de Computação de 2008, da qual fiz parte, por ter me aturado ao longo desses 5 anos, pelas brigas, discussões e bagunças, tenho certeza que conheci as pessoas com linhas de pensamento evidentemente diversificadas. Certamente foi gratificante estarmos juntos ao longo dessa jornada. Valeu moçada!

RESUMO

Um dos maiores desafios de robótica móvel autônoma é a complexa tarefa de navegar em um ambiente desconhecido e capturar informações do mesmo, construindo um mapa e se localizando. Para desenvolver um algoritmo que proporcione a um robô móvel a capacidade de mapear e localizar em um ambiente desconhecido é necessário que o robô adquira e se utilize de informações extraídas para mapear o ambiente através de odometria, sensores de toque e sonar, entre outros, os quais o auxiliam na identificação e mapeamento obstáculos, respondendo simultaneamente a essa interação com o ambiente. Porém, a estimativa da posição do veículo é incerta em razão de erros acumulados já, conhecidos da odometria. O método de localização proposto é baseado em mecanismos de estatística, através de comparações dos dados colhidos por um sonar com o conhecimento que o robô tem de seu ambiente.

Palavras-chave: Monte Carlo, Localização, Mapeamento, Robôs móveis.

ABSTRACT

One of toughest challenges in autonomous mobile robotics is the complex task of navigation in unknown surroundings, capturing information, setting up a map and localizing itself within it. An algorithm that provides a mobile robot the ability to map and locate itself in an unfamiliar environment must acquire and use the extracted information to map the environment through sensors like odometry, touch and sonars, which aid in identifying and dodging obstacles, responding in real time to interactions with the world. However, the approximate vehicle?s position is uncertain due to accrued errors in odometry. The proposed localization method is based on statistical mechanisms, using comparisons of data harvested by a sonar with the robots knowledge about its environment.

Key-words: Monte Carlo, Location, Mapping, Mobile robots

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABS - Anti-lock Braking System

API - Application programming interface

ARM - Acorn RISC Machine / Advanced RISC Machine

GPS - Global Positioning System

JVM - Java Virtual Machine

LCP - LEGO Communications Protocol

MCL - Monte Carlo Localization

MIT - Massachusetts Institute of Technology

NXT - Next

PC - Personal Computer

RCX - Robotic Command Explorer

SLAM - Simultaneous Localization and Mapping

USB - Universal Serial Bus

VM - Virtual Machine

LISTA DE SÍMBOLOS

α - Ângulo de scanner do sonar

β - Ângulo de orientação da partícula

X - Eixo horizontal *x*

Y - Eixo vertical *y*

x - Coordenada *x*

y - Coordenada *y*

Dist. - Distância entre dois pontos

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação do tema

Qualquer automatização de uma tarefa executada por um robô móvel requer o conhecimento de sua localidade. Robôs móveis que operam em ambientes fechados e não estruturados não contam com auxílios, como GPS para determinar a sua localização. A utilização dos sensores de odometria disponíveis em suas rodas, para o mapeamento e localização, é imprecisa e pode acumular erros ao longo do tempo, o que torna inviável a sua utilização isolada para esse fim. Portanto, para este tipo de ambiente, torna-se necessária a implementação de algoritmos que utilizam dados adicionais dos sensores disponíveis e os integram de forma a obter uma estimativa adequada de sua localização (THRUN; BüCKEN, 1996). Neste trabalho é descrito um algoritmo de localização que utiliza métodos probabilísticos,através de encoder, para o mapeamento e a aplicação do Método de Monte Carlo, que estima simultaneamente a localização do robô.

1.2 Objetivo

O objetivo deste trabalho é implementar um algoritmo de localização de robôs móveis que contemple as características do estado da arte, dentro das limitações dos sensores disponíveis para o trabalho; consequentemente, este método implica em uma navegação robusta em tempo real, conforme levantamento realizado em pesquisa da literatura mais atual da área. Esta abordagem evita a criação de **landmarks** artificiais no ambiente em que o robô atua.

1.3 Justificativa

Além de ser uma área de pesquisa desafiadora, que apresentou ao autor desta monografia grandes oportunidades para sua evolução na pesquisa científica, visando a preparação para a continuação dos estudos com término da graduação, em nível de mestrado, a localização de robôs móveis são de importância fundamental para o cumprimento dos objetivos do grupo de iniciação científica em robótica Reaction, fundado pelo autor deste trabalho, com atuação na UNIARA, e que prevê a implementação de um sistema de navegação completo para a participação em congressos e competições de robótica, e a apresentação de métodos de navegação para

robôs que são desprovidos de informações do ambiente.

1.4 Problema e hipótese de pesquisa

Produzir um algoritmo de localização de robôs móveis para um protótipo de sistema de navegação, que servirá de prova de conceito e permitirá avaliar e contabilizar diversas métricas de performance de localização.

1.5 Metodologia

Para o desenvolvimento deste trabalho foi realizada uma revisão bibliográfica acerca de algoritmos, equipamentos e sensores, necessária para a realização da tarefa de localização de robôs móveis. Implementar um algoritmo de localização de robôs móveis utilizando os recursos computacionais e de hardware disponíveis na UNIARA. Integrar esse algoritmo em um sistema de navegação para robôs móveis para a realização de testes e coletar dados para a análise dos resultados. Realizar uma análise dos resultados com base em métricas que foram definidas durante o trabalho, e realizar análises comparativas com outros trabalhos disponíveis na literatura. Documentar os resultados obtidos e publicar artigos.

1.6 Cronograma

Etapa	Meses											
	01	02	03	04	05	06	07	08	09	10	11	12
Revisão Bibliográfica	X	X	X	X	X	X	X	X	X	X	X	
Escolha da Tecnologia				X	X	X	X					
Prova de Conceito					X	X	X	X	X			
Documentação dos resultados									X	X	X	

Tabela 1: Cronograma de Atividades 2012

2 CONCEITOS DE ROBÓTICA

2.1 Robôs móveis

Um robô nada mais é do que um conjunto de dispositivos eletromecânicos ou biomecânicos que realiza tarefas atrás de um controle humano, de maneira pré-programada ou autônoma. No decorrer dos últimos anos e com o constante desenvolvimento tecnológico, sistemas robóticos móveis e fixos passaram a integrar ambientes cotidianos. Inicialmente desenvolveu-se a robótica industrial (robôs fixos), que atuam em ambientes estruturados, predefinidos e praticamente estáticos o que facilita o seu controle, uma vez que esses ambientes têm seus objetos em posições definidas, como uma indústria automobilística. Posteriormente, a robótica móvel esteve um crescimento considerável introduzindo a movimentação e a interação com o ambiente altamente dinâmico, ambiente este que tem seus conteúdos móveis independentes do robô, aumentando assim, a complexidade de desenvolvimento dessa categoria. Robôs desse porte têm uma grande preocupação com a sua localização, pois sofrem com as variações do ambiente, gerando erros significativos que afetam a execução de suas tarefas. Esse problema não se encontra em robôs fixos, pois estes sabem exatamente onde estão por terem uma base fixa. Contudo, robôs móveis necessitam de um sistema inteligente e de uma constante interação com o meio, tendo reações às suas percepções sem afetar seus resultados.

2.1.1 Robôs com rodas

Figura 1: Robô Pioneer (Google Imagens)

Robôs com rodas navegam na extensão terrestre, são de fácil fabricação, programação e de baixo custo. Porém, com as rodas, a movimentação sobre obstáculos é difícil, como em terrenos rochosos e acentuados. É o mais popular dos robôs e certamente entrará primeiro no mercado, pois a sua aplicação é bem eclética e próxima de nossas realidades. Robôs semelhantes a esses são utilizados pela polícia para o desarmamento de bombas, quando o risco de explosão e,

consequentemente, o risco de vida, são considerados altamente críticos. A figura 1 ilustra um robô com rodas, o Pioneer, muito utilizado nos centros acadêmicos, dotado de rodas e um sensor.

2.1.2 Robôs terrestres(rasteiros)

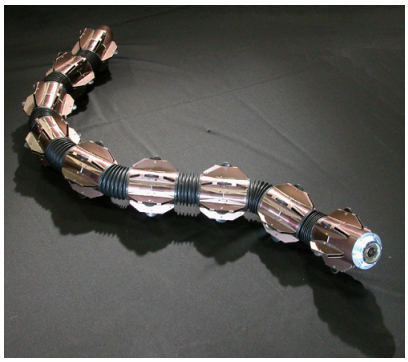

Figura 2: Snakebots da Nasa (CRACKED, 2010)

Com a capacidade de se mover como cobras biológicas, o *snakebot* é muito utilizado para combater incêndios pela SINTEF. A faculdade de Carnegie Mellon University desenvolve esse robô para a medicina, fino o suficiente para manobrar dentro dos órgãos humanos, possibilitando diagnósticos detalhados e precisos, ou ainda operações de grande risco. Pode mover-se em diferente ambientes por conta da sua facilidade em mudar de formas. A figura 2 faz parte das últimas gerações de sondas robóticas que a NASA planeja desenvolver, a fim de explorar outros planetas.

2.1.3 Robôs com pernas

Figura 3: Robô Aranha (Google Imagens)

Excelente para terrenos irregulares, porém com grande dificuldade de locomoção e lentidão, pode auxiliar na recuperação de deficientes físicos, idosos, e na exploração de planetas desconhecidos pela humanidade. O robô inspirado em uma aranha, mostrado na figura 3 demonstra

que essa categoria pode ter diversas quantidades de pernas. O desenvolvimento e o controle de inteligência artificial, por trás desses quadrúpedes, devem ser levadas em consideração pela sua alta complexidade.

2.1.4 Robôs aéreos

Figura 4: Robô helicóptero controlado por iPhone (KINGSLEY-HUGHES, 2010)

Deslocando-se tridimensionalmente, os robôs aéreos são capazes de percorrer ambientes externos ou internos através de voo com propulsão ou flutuação. São muito úteis no monitoramento aéreo para o combate de incêndios, a localização e o resgate de pessoas em áreas de difícil acesso, como lugares montanhosos e de relevo acidentado, patrulha de fronteiras e também na coleta de amostras na atmosfera. Seus estudos também são aplicados para compreender melhor o esquema de voo e comunicação de pássaros biológicos. A figura 4 ilustra o Drone, *"brinquedo"* controlado por iPhone; apesar da complexidade apresentada anteriormente, este é de simples manipulação.

2.1.5 Robôs aquáticos

Figura 5: Robô peixe do MIT (MIT, 2009)

O MIT desenvolveu o peixe robô, como mostrado na figura 5, com a finalidade de ter acesso fácil a locais subaquáticos onde veículos tradicionais tripulados não poderiam alcançar; também podem ser usados para inspecionar vazamentos de canos de gás e petróleo. A Petrobras tem investido em pesquisa nessa categoria de robôs, uma vez que ela é pioneira na exploração de petróleo em alto-mar.

2.2 Localização e mapeamento

Localização é um problema comum na navegação de robôs autônomos, que está relacionado com a capacidade do robô em conhecer a sua posição e, simultaneamente, o mapeamento relacionado à capacidade de conhecer as estruturas que se encontram ao redor do robô. O fornecimento de um mapa ao robô nem sempre é uma abordagem confiável, pois o ambiente pode sofrer mudanças no decorrer da navegação; o mesmo conceito é válido para a localização. Com este cenário, é esperado que o robô deva ser capaz de estimar a sua localização, baseada em um mapa gerado pelo seus próprios sensores.

Segundo (BOAS, 2011) as informações geradas pelos sensores podem possuir erros por causa das limitações físicas dos mesmos, ou por fenômenos que não podem ser modelados, devido a sua complexidade ou imprevisibilidade. Esses erros fazem com que as estimativas geradas pelo robô sejam imprecisas, limitando assim, a utilização em algoritmos posteriores que dependam de resultados mais precisos de mapeamento e localização.

O SLAM *Simultaneous Localization and Mapping* (mapeamento e localização simultâneos) é um grande desafio da robótica móvel. A dificuldade de iniciar a construção de um mapa, a partir de um ponto desconhecido do ambiente em que está inserido o robô, exige utilizar informações incertas extraídas a partir de sensores e, simultaneamente, usar o mapa para localizar-se. (WIJESOM, 2006)

Existem dois tipos de localização: local e global. A local foca-se em compensar erros de odometria (método para estimar a posição de um robô) durante navegação e necessita de uma localização inicial conhecida. Já a global localiza sem conhecer seu ponto de origem e é capaz de lidar com problemas como sequestro, em que um robô é transportado fisicamente para algum lugar desconhecido, sem qualquer informação; a recuperação pode acarretar em graves erros de localização.

Segundo (THRUN et al., 2000), o processo de localização do robô em relação a uma determinada posição do ambiente é chamada de posição estimativa ou rastreamento. A localização do robô é importante, pois quase todas as tarefas da robótica requerem essa informação. Ainda segundo Thrun, a localização pode ser nesta como um problema de transformação de coordenadas e mapas são um sistema de coordenadas global, que é independente da representação do robô. A localização é um processo de estabelecimento e de correspondência entre o sistema de coordenadas globais e coordenadas de locais.

Com a localização e mapeamento, estamos interessados em estimar a posição exata do robô em tempo real.

2.3 Robótica baseada em comportamento

A biologia sempre serviu de inspiração para a robótica baseada em comportamento, pois trouxe diversas formas de controle que em contextos ambientais geram resultados complexos e eficientes. Essa metodologia é baseada em um conjunto de módulos com objetivos bem desafiadores, como o de desviar de um obstáculo, evitando colisões. Um observador externo interpreta os comportamentos como padrões de atividades do robô, emergentes das interações entre o robô e seu ambiente. Já um programador vê os comportamentos como módulos de controle com funções bem definidas e limitadas, que trabalham em conjunto a fim de alcançar uma meta desejada. (SANTERIO, 2010)

Para Arkin, a melhor definição de robôs comportamentais vem da psicologia: *"um comportamento, simplesmente, é uma reação a um estímulo"*. Nada mais são que agentes inteligentes com capacidade de interação com o mundo físico, consequentemente, suas atitudes são diretamente influenciadas pelo meio. Através de sensores (visuais, sonoros e táteis) é possível interpretar o mundo real, podendo definir uma estratégia para concluir a tarefa. Porém esse projeto não é tão simples quanto aparenta, pois sua complexidade pode aumentar uma vez que os ambientes contenham "ruídos"e sejam totalmente imprevisíveis e dinâmicos. Robôs comportamentais dependem de um bom sistema de navegação e localização para alcançar com sucesso a execução de seu objetivo.

Em ambientes reais existe uma infinidade de fenômenos com os quais um robô baseado em comportamento pode interagir, porém a riqueza resultante emergente está totalmente relacionada a essa riqueza de detalhes do ambiente. A figura 6 ilustra a complexidade de interação e o comportamento emergente em função do ambiente. Esses robôs demonstram uma característica de emergir a partir de interações.

2.4 Método de Monte-Carlo

O método de Localização de Monte Carlo localiza o robô globalmente usando um algoritmo de condensação baseado na estatística. Através das informações dos sensores, é modelado um conjunto de probabilidades da sua localização; cada probabilidade ou hipótese se refere a uma posição no mapa. São realizadas comparações desses possíveis pontos com as informações extraídas do robô e, com essas comparações, são dados pesos para cada uma das hipóteses. Ao mover-se, o robô faz novas coletas de informações ambientais e apura melhor as suas probabilidades, dando a elas novas pontuações, consequentemente, existiram partículas com pontos maiores que outras, aumentando a sua probabilidade; sendo assim, a quantidade de hipóteses

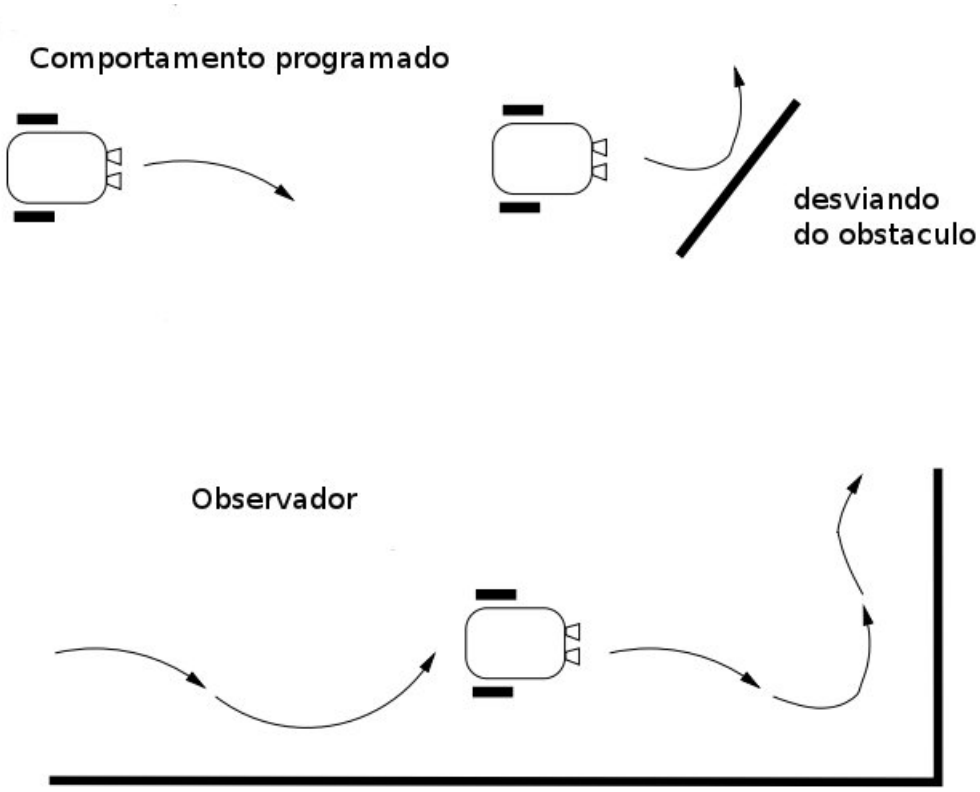

Figura 6: Comportamento emergente (ARKIN, 1998)

diminui.(FOX et al., 1999)

A movimentação de robôs móveis no mundo real requer uma precisão sobre a sua localização. O MCL baseia-se na probabilidade e utiliza um mapa baseado no ambiente, que foi extraído com a movimentação do robô através dos sensores.

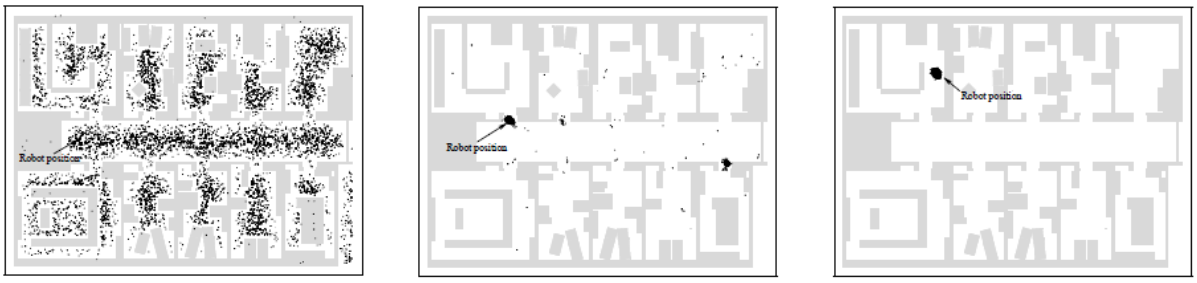

Figura 7: Exemplo da utilização do método de Monte Carlo (THRUN et al., 2000)

A figura 7 ilustra o MCL na prática, mostrando uma coleção de hipóteses (representada em 2D) geradas durante a localização global de um robô orientado por um sonar. No primeiro passo, a posição do robô é globalmente incerta e as amostras são distribuídas uniformemente através do espaço livre. Já a segunda figura mostra o conjunto de amostra depois de aproximadamente 1 metro de movimento do robô, ponto em que o MCL não tem ambiguidade de posição. Por fim, no último passo e após 2 metros de deslocamento do robô, a ambiguidade das

hipóteses foi resolvida. Nesse passo o robô sabe qual a sua localização. As maiorias das amostras estão agora centradas firmemente em torno da posição correta, como mostrado no ultimo passo. (THRUN et al., 2000)

Uma segunda dimensão, que tem um impacto substancial sobre a dificuldade de localização é o meio ambiente. Os ambientes podem ser estáticos ou dinâmicos.

Ambientes estáticos: Todos os outros objetos nos ambientes permanecem no mesmo lugar para sempre, sem qualquer tipo de movimentação ou modificação por terceiros. Têm algumas boas propriedades matemáticas que se tornam passíveis de estimativa probabilística eficiente.

Ambientes dinâmicos: Possuem outros objetos que o robô percebe a sua movimentação ou alteração de posição ao longo do tempo. De modo particular o interesse nas mudanças que persistem ao longo do tempo. Alterações que não são mensuráveis, evidentemente, e de nenhuma relevância para a localização, e aqueles que apenas afetam uma única são melhor tratadas como ruído

Obviamente, a localização em ambientes dinâmicos é mais difícil do que a localização em estáticos. (THRUN et al., 2000)

2.5 LEGO Mindstorms NXT

Em 2006 a LEGO apresentou ao público o **Mindstorms NXT**, uma versão aprimorada do seu primeiro kit educacional de robótica, o módulo RCX (Robotic Command Explorer), resultado de uma parceria com o Media Lab do MIT. Assim como o tradicional LEGO, o Mindstorms baseia-se em um conjunto de peças (pertencentes a LEGO Technic) que se encaixam, permitindo inúmeras combinações de layouts e montagens. Para o enriquecendo do kit, existem diversos aditivos de sensores, motores e uma CPU (RCX ou a nova versão NXT) que modificam e abrem um leque de arranjos a ser explorado, possibilitando a criação de robôs simples, passíveis de executar operações pré-programadas. O kit utilizado para o desenvolvimento da prática deste trabalho foi o **Mindstorms NXT 8547**, composto por 431 peças, como ilustrado na figura 8.

O único propósito de utilizar esse kit é minimizar o tempo gasto para o desenvolvimento físico de um robô, evitando problemas de eletrônica, sem mencionar a flexibilidade que o kit oferece em elaborar diversos layouts.

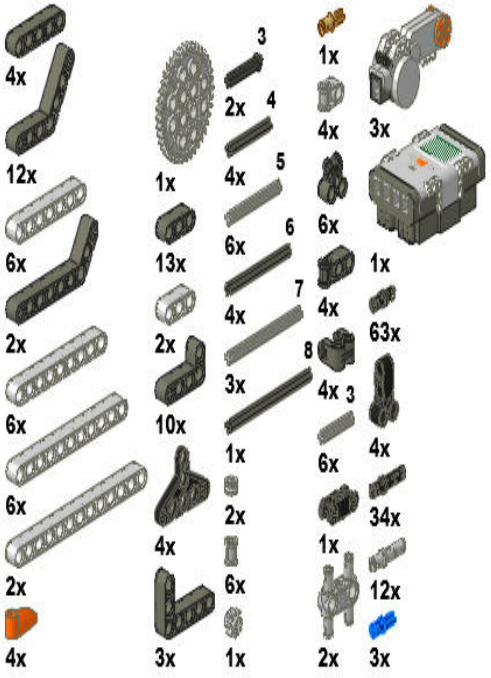

Figura 8: Peças do Kit Mindstorms NXT 8547 (Google Imagens)

2.5.1 Especificação de hardware do NXT

O NXT é um *"Programmable Brick"*, um dispositivo capaz de interagir com eventos externos do mundo real através de sensores, movimentar-se com os atuadores e também receber feedbacks a partir destes dispositivos.

O tijolo como é popularmente conhecido, é totalmente programável; basta adicionar os sensores, atuadores e peças que desejar para iniciar o desenvolvimento. Para aumentar a sua flexibilidade e abrangência, o bloco é composto por:

- 1 processador principal (Atmel 32-bit ARM);
- 1 co-processador (Atmel 8-bit AVR processor);
- 1 porta de comunicação USB 2.0 (12 Mbit/s);
- 3 portas de saída digital;
- 4 portas de entrada (uma IEC 61158 Type 4/EN 50170 compliant);
- 4 botões de operação para o usuário;
- Display LCD preto e branco (100 x 64 pixels);

Figura 9: LEGO Mindstorms NXT (SOLORZANO, 2012)

- Alto-falante com 8-bit de resolução;
- Bateria recarregável de lítio;
- Bluetooth (CSR BlueCoreTM 4 v2.0 +EDR).

O brick permite que arquivos de som possam ser baixados usando uma porta USB ou Bluetooth, e que possam ser copiados entre dois blocos NXT por Bluetooth. Simultaneamente, pode se utilizar até três conjuntos NXT via Bluetooth.

2.6 LeJOS NXJ

A versão original do kit fornece um software para escrever programas visualmente (baseado no LabVIEW), permitindo que o programador, ou até mesmo uma criança consiga desenvolver programas utilizando fluxogramas em vez de de linhas de código.

Porém não existe uma restrição ou trava à sua plataforma de desenvolvimento; o NXT permite a troca do seu firmware, que é um conjunto de operações programadas diretamente no hardware, e oferece uma vasta gama de linguagens de programação, como demonstrado na tabela 2.

O leJOS é um projeto open sourcer hospedado no repositório SourceForge, criado a partir do TinyVM, que fornece um ambiente de programação Java para o NXT com uma substituição de firmware original do Mindstorm. O firmware inclui uma JVM, uma biblioteca de classes

Linguagem	API	Link
C#	Microsoft Robotics Studio	*http://www.microsoft.com/robotics/*
C/C++	nxtOSEK	*http://lejos-osek.sourceforge.net/*
C	NXTGCC	*http://nxtgcc.sourceforge.net/*
Nativo do NXT	Robolab	*http://www.legoengineering.com/programming.html*
Java	leJOS	*http://lejos.sourceforge.net/*
LabVIEW	nxtmastery	*http://nxtmastery.com/*
Python	NXT-Python	*http://code.google.com/p/nxt-python/*
Ruby	ruby-nxt	*http://rubyforge.org/projects/ruby-nxt/*

Tabela 2: Firmwares suportados pelo NXT

que implementam API, ferramentas para fazer upload do binário a ser executado no NXT, uma API PC para comunicar com o NXT através de Bluetooth, USB ou LCP. A API oferece uma biblioteca que suporta operações bem abstraídas, como a de navegação, mapeamento e comportamento que foram o controle de todos os sensores, até mesmos aqueles não presentes nesse kit. Esses isolamentos têm recebido uma maior atenção nas últimas versões.

Figura 10: Firware leJOS (SOLORZANO, 2012)

Até a conclusão do desenvolvimento deste trabalho, o leJOS esta na versão 0.9.1, disponível para Microsoft Windows e Linux / Mac OS, e que foi a utilizada para o desenvolvimento deste trabalho. A equipe de José Solórzano, criador do leJOS, desenvolveu um plug-in para o IDE Eclipse, o que facilita no desenvolvimento e no deploy do código.

Segundo o tutorial (SOLORZANO, 2012) a API é constituída por:

- Firmware substituto para o NXT, que inclui uma máquina virtual Java;
- Uma biblioteca de classes Java (classes.jar) que implementam o lejos NXJ;
- A API PC para escrever programas que se comunicam com o leJOS usando Bluetooth ou USB ou LCP.
- Muitos programas de exemplo.

É importante salientar que na substituição do firmware original para o leJOS, serão apagados todos os arquivos atualmente detidas no firmware LEGO ou outro que esteja instalado no bloco. O firmware LEGO pode ser restaurado usando o software fornecido pela LEGO.

3 PROCESSO DE DESENVOLVIMENTO DO ROBÔ

A motivação da construção física do robô é a de colocar em prática em execução o algoritmo de localização que foi desenvolvida. "O projeto da parte mecânica do sistema robótico é fundamental para o bom funcionamento global do conjunto. Devem ser considerados diversos fatores relativos às necessidades específicas de cada sensor, cada atuador e cada manipulador."(SOUSA, 2003) Nessa fase alguns pontos são importantes e que influenciaram diretamente na implementação do algoritmo, como a escolha dos sensores e a definição do tipo de locomoção, posicionamento de sensores, etc. A tomada de decisão para esses pontos pode aumentar ou diminuir a complexidade de entendimento e desenvolvimento da inteligência artificial do robô e até mesmos seus comportamentos. Tendo em mente todos os adendos para a construção mecânica do robô móvel e o tempo gasto para a sua elaboração e fabricação, foi optado por desenvolver e utilizar nesse trabalho o kit da LEGO, pois ele facilita a manutenção do foco no desenvolvimento do algoritmo, em vez de criar preocupações com problemáticas de hardware e mecânica que não são os objetivos deste trabalho e, consequentemente otimiza-se o tempo.

3.1 Estrutura mecânica do protótipo

Encontrar um modelo ideal ou próximo dele exige uma análise de todos os possíveis cenários de atuação e as necessidades de informações que o robô necessita. Portanto, projetar uma versão final de um protótipo exigiu uma gama de testes, por tentativa e erro, que resultaram em algumas versões dos androides. Cada versão apresentou uma limitação física que certamente dificultaria o desenvolvimento de seu core.

As quatro versões desenvolvidas, demonstradas na figura 11, apresentaram dificuldades de direção o que iria prejudicar na odometria; a não movimentação horizontal do sonar também se apresentou como ponto negativo, pois é interessante utilizar este sensor como scanner. Por conta dessas limitações, as quatro versões desenvolvidas foram descartadas.

Após algumas tentativas não satisfatórias dos protótipos montados com o mindstorm, houve a necessidade de recorrer a modelos prontos. Surgiram então algumas opções já desenvolvidas e elaboradas que são oferecidas gratuitamente em portais pela internet, entre tantos o portal *nxtprograms.com* é o que oferece uma maior quantidade de modelos e fornece um tutorial passo a passo da construção de cada um deles.

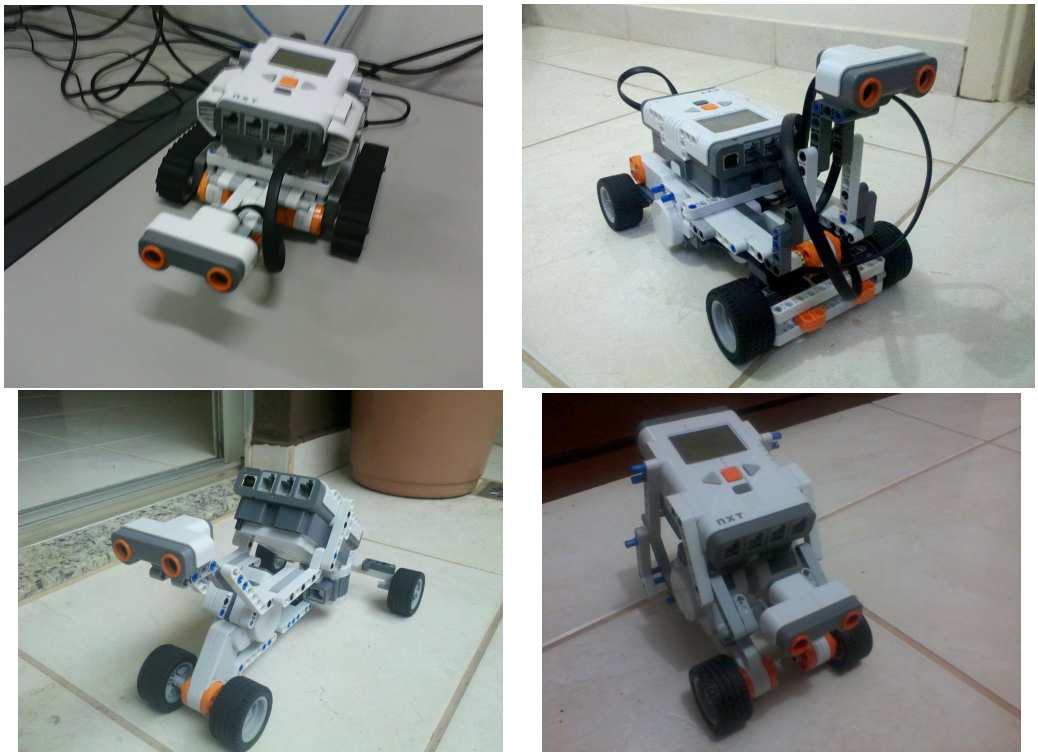

Figura 11: As quatro primeiras versões do protótipo

Dave Parker, treinador e educador de materiais da LEGO Mindstorms NXT, oferece em um de seus portais, nxtprograms.com, soluções físicas (acompanhadas de tutoriais de construção) para diversas utilizações e finalidades do kit da LEGO.

Além de oferecer soluções físicas, as mesmas são acompanhadas de uma solução de sotfware. Contudo, as mesmas não foram utilizadas, uma vez que o objetivo desse trabalho é desenvolver e implementar um desfecho de localização. Vale salientar que as tecnologias utilizadas nos softwares oferecidos por Paker são distintos daqueles optados neste trabalho. Parker utiliza do lab view, ambiente nativo do kit da LEGO.

Dentre as soluções físicas oferecidas, duas foram escolhidas: **Race Car**, como ilustrado na figura 12; e **Explorer**, como apresentado na figura 13. Por apresentar duas características fundamentais o modelo Explorer foi a escolhido,, por fornecer uma rotação sob um eixo Y, assim como o planeta Terra, e por movimentar um de seus sensores (sonar), o que é uma grande vantagem para o desenvolvimento.

A solução oferecida pelo Race Car apresentou desvantagens em relação ao Explorer: primeiro, por não rotacionar sob seu eixo vertical; segundo, por ter a necessidade de adaptação do modelo, pois o mesmo não é dotado de sensores, e esses são fundamentais para o nosso problema. Contudo, a solução escolhida para este trabalho foi o Explorer, já que a sua maior vantagem em relação aos modelos e protótipos apresentados está na movimentação horizontal

Figura 12: Race Car
(PARKER, 2012)

do sonar, o que proporciona a utilização do mesmo com scanner, agregando grande valor à solução.

Figura 13: Explorer
(PARKER, 2012)

O Explorer é composto por dois sensores, um ultrasonic (ultrassônico) e um touch (toque) representados respectivamente na figura 20. O ultrasonic é utilizado para medir a distância que o robô está de um obstáculo; essa tarefa é executa através da geração de ondas sonoras de alta frequência imperceptíveis aos ouvido humano. Ao colidir com um objeto a onda é refletida retornando ao sensor, que por sua vez calcula a distância a partir do tempo de resposta. Este mesmo mecanismo é utilizado na orientação e navegação de submarinos. Já o touch trabalha na detectação por pressão sobre ele, um simples toque mecânico.

A plataforma móvel escolhida e utilizada, apresentada na figura 13, movimenta-se através

de duas rodas dianteiras cada um delas com um *encoder* que fornece as coordenadas cartesianas no decorrer da sua navegação e uma terceira roda e, esta traseira, que simplesmente mantém o equilíbrio do robô girando livremente.

4 PROCESSO DE DESENVOLVIMENTO DA INTELIGÊNCIA ARTIFICIAL

Sistemas robóticos são soluções com algum grau de liberdade em suas decisões e controles. Para isso existe uma grande interatividade com o ambiente e, possivelmente, com sistemas periféricos. Projetar e desenvolver um algoritmo dessa natureza é uma tarefa extremamente complexa, que exige um domínio de diversas áreas da computação e da matemática, como: programação, sistemas distribuídos, inteligência artificial, estatística, álgebra linear e geometria analítica. Como em qualquer outro projeto, a identificação de requisitos iniciais é fundamental para o e essa tarefa consiste em analisar as características e metas a serem atendidas.

4.1 Sistemas operacionais inteligentes de tempo real

Sistemas de tempo real são classificados como sistemas reativos que interagem com respostas contínuas de estímulos, provocados pelo seu ambiente ou por um agente, e estão diretamente relacionadas com o ambiente inserido e com o seu comportamento temporal. A principal característica desses sistemas não é a integridade e a consistência de suas respostas, mas principalmente o prazo em que são obtidas. O prazo é o principal requisito não funcional trabalhado nesses sistemas. A cada reação o sistema deve entregar resultados corretos e precisos em um tempo especificado; o não comprimento desse prazo e da integridade dos dados pode ocasionar danos catastróficos. Nesse nicho de sistemas, o estouro do limite temporal é considerado como falha (FARINES JONI DA SILVA FRAGA, 2000). Sistemas com essa característica são classificados como sistemas de tempo real críticos, cuja violação do prazo para a execução da tarefa deve ser respeitado e cumprido, como o acionamento de um freio ABS ou a movimentação do joystick de um avião, que devem ter respostas praticamente instantâneas.

Na construção de sistemas robóticos autônomos o item de maior importância está relacionado ao desempenho temporal. Para se obtiver resultados positivos e satisfatórios com o sistema é fundamental definir as tarefas e comportamentos antes de iniciar a implementação. Em um escopo com múltiplas informações, é imprescindível a elaboração de um sistema *multi-threads*, um ambiente multitarefa diferentemente de um sistema de processos. "Processos operam com espaços de endereçamento diferentes entre si, enquanto threads compartilham o mesmo espaço de endereço, além de vários outros recursos."(LORA ELDER M. HEMERLY,)

O desenvolvimento do algoritmo esta estruturado em multi-thread, porém o objetivo não é

explorar o poder computacional do ARM disponibilizado pelo NXT, e sim ter diversos comportamentos e tarefas distintas sendo executadas simultaneamente, surgindo assim às primeiras dificuldades do algoritmo, considerando as limitações de processamento e memória do ARM.

4.2 Arquitetura da inteligência

A arquitetura desenvolvida, representada conforme a figura 14 que descreve um diagrama de instalação, com a representação dos componentes e seu envolvimento com outros dispositivos de suporte ao processamento.

A arquitetura é dividida em três módulos:

- **NXT:** módulo móvel, dispositivo que interage diretamente com o ambiente.
- **Server:** módulo fixo, proprietário de toda a informação do ambiente, suas decisões influenciam diretamente no comportamento do androide.
- **Comunicação:** módulo compartilhado entre os outros dois, fornecendo todo o canal de comunicação e a troca de dados entre eles. Os dados são transmitidos entre os módulos via Bluetooth.

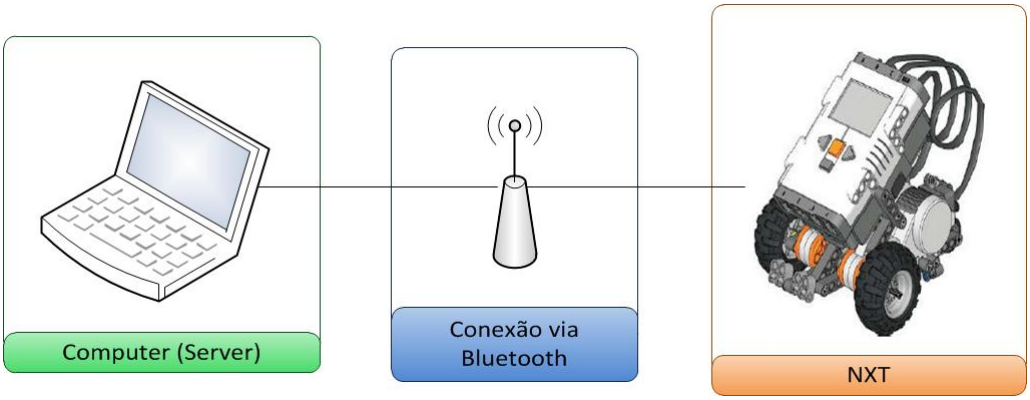

Figura 14: Diagrama de Comunicação

Desenvolver aplicações para sistemas embarcados requer certos cuidados que devem ser levados em consideração como limitações do hardware, tais como processamento, memória e energia, que levaram o autor deste trabalho a transferir o processamento "pesado"de localização e mapeamento para um hardware que fornecesse maior poder computacional. Contudo, a transmissão de dados deveria ser feita através de um protocolo no qual o Bluetooth atende expectativas apesar de ter uma área de cobertura pequena em relação a outro protocolos.

4.2.1 Componente NXT

A arquitetura proposta para o módulo NXT é constituída por: Coletor de dados, Controle da mobilidade e Controle de Navegação. Este módulo é responsável por coletar e interpretar os dados do ambiente. As informações coletadas podem disparar duas ações assincronamente, o despache dos dados para o Server ou uma movimentação física do dispositivo.

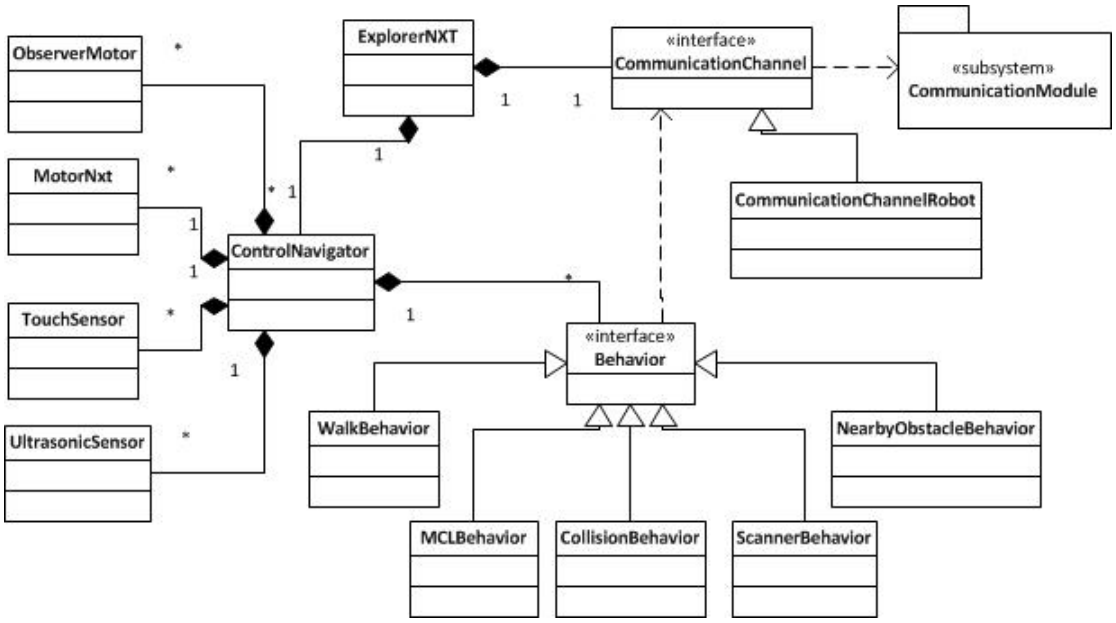

Figura 15: Diagrama de classe do módulo móvel

- **Coletor de dados:** Através dos sensores é possível fazer uma leitura do ambiente; todos os dados coletados por esses sensores são filtrados, repassando somente o que é pertinente ao controle de navegação, comportamentos, mapeamento e localização, que resultarão em ações executadas pelos atuadores. Existem três agentes de coleta de dados, separados verticalmente (rodas, touch e sonar).

- **Controle de mobilidade:** As rodas não funcionam simplesmente como meios de locomoção para movimentar fisicamente o robô de um ponto ao outro, mas também são utilizados como um simples localizador GPS, já que por odometria é possível localizarmos a posição do robô baseados em coordenadas cartesianas. Por ter um eixo engrenado e um girando em falso, é possível para o robô rotacionar sobre seu próprio eixo.

- **Controle de navegação:** É o grupo responsável pela coleta dos dados lidos pelos sensores e através, deles, analisar e planejar rotas de fuga de obstáculos. O sistema de navegação é totalmente autônomo, baseado em comportamentos. Os comportamentos têm caráter reativo, lidando com as mudanças dinâmicas do ambiente em tempo real.

O diagrama de classe apresentado na figura 15 compõe o módulo móvel, no qual existe uma dependência com um subsistema de comunicação para o tráfego de dados com o módulo servidor através da interface *CommunicationChannel*, que é implementada pela classe *CommunicationChannelRobot*. Este módulo é composto por classes que atuam diretamente na execução e no comportamento dos atuadores de navegação (*MotorNXT*), deslocando fisicamente o robô e o de observação (*ObserverMotor*) que movimenta fisicamente na horizontal o sonar, que é gerenciado pela classe *UltrasonicSensor*. O *TouchSensor* controla os dados do sensor de toque. Já as classes que implementam a interface *Behavior* definem o comportamento do robô, que por sua vez solicita requisições ao motores deste androide.

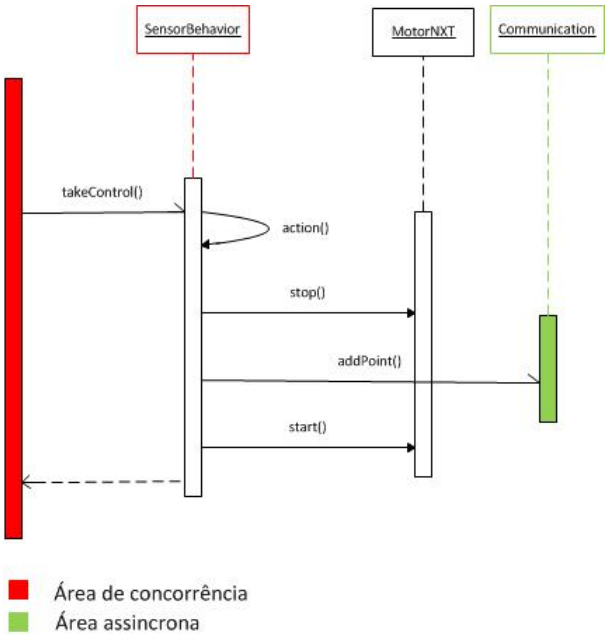

Figura 16: Diagrama de sequência do módulo móvel

Na identificação de um obstáculo, por qualquer um dos sensores, o *takeControl()* é ativado iniciando o comportamento correspondente ao sensor que notificou o encontro do obstáculo, invocando a ação do comportamento (através da chamada *action()*). Cada Behavior terá um tratamento diferente; de modo geral o movimento do robô é paralisado para a coleta de dados. Esses dados são adicionados assincronamente em um fila do canal de comunicação ativando o movimento do robô. Os dados armazenados na fila de comunicação do NXT são enviados para o Server e, após o envio, os dados são apagados da fila a fim de preservar e disponibilizar memórias no ARM do NXT. Conforme ilustrado no diagrama de sequência pela figura 16, existem pontos críticos como a área de concorrência na identificação de obstáculo. O tratamento para este caso é o nível de prioridade, pois o comportamento identificado pelo sonar tem prioridade sob o sensor de toque. A área destacada em verde pelo diagrama de sequência demonstra o ponto de assincronismo, pois tanto a notificação de obstáculo, quanto os dados da odometria,

são enfileirados na estrutura de comunicação.

4.2.2 Componente Server

É de responsabilidade da torre de controle (servidor) se comunicar com o dispositivo móvel (robô) via bluetooth, a fim de extrair informações do ambiente em que o NXT está presente. Com esses dados o Server deve gerar e atualizar um mapa, representando os obstáculos e a localização do robô em tempo real. Com base nos dados recebidos e com o histórico de navegação, o Server deve estimar a localização do robô utilizando o algoritmo a ser desenvolvido neste trabalho.

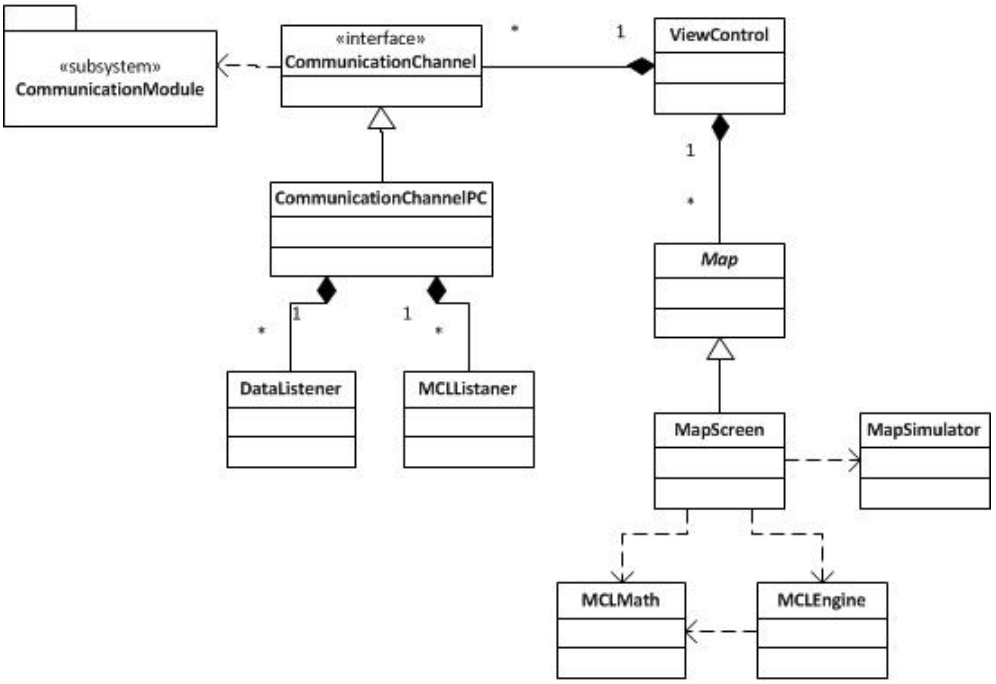

Figura 17: Diagrama de classe do módulo server

Conforme apresentado pela figura 17, o diagrama de classe demonstra a organização do módulo servidor, que também depende de um subsistema de comunicação com o módulo móvel. Existe uma sutil diferença, na camada de comunicação, entre os módulos móvel e server, que a presença de Listener notifica o recebimento de novos dados. O *DataListener* é o responsável pela construção do mapa baseado nos dados enviados pelo robô, conforme exemplificado na seção anterior. Já o *MCLListener* é o que dispara as chamadas para os cálculos do método de Monte Carlo através da classe *MCLEngine*. As notificações percebidas por ambos os listener geram resultados apresentados no mapa, que são gerenciados pela *MapScreen*. O *MapSimulator* permite a simulação do algoritmo de Monte Carlo através de parâmetro e não está vinculado com qualquer comportamento ao robô.

Para o Server a arquitetura proposta é composta por:

- **Tradutor de dados:** Ao receber os dados vindos do NXT, os mesmos são inseridos em uma pilha de dados a fim de criar um histórico.
- **Gerador de mapa:** Com uma leitura constante, os dados do histórico são verificados em busca de novidades para a atualização e construção do mapa em que o NXT se encontra.
- **Processador de localização:** Baseado em um conjunto de amostras retiradas do ambiente e utilizando métodos matemáticos, estimar com precisão a localização do robô não é uma tarefa complexa.

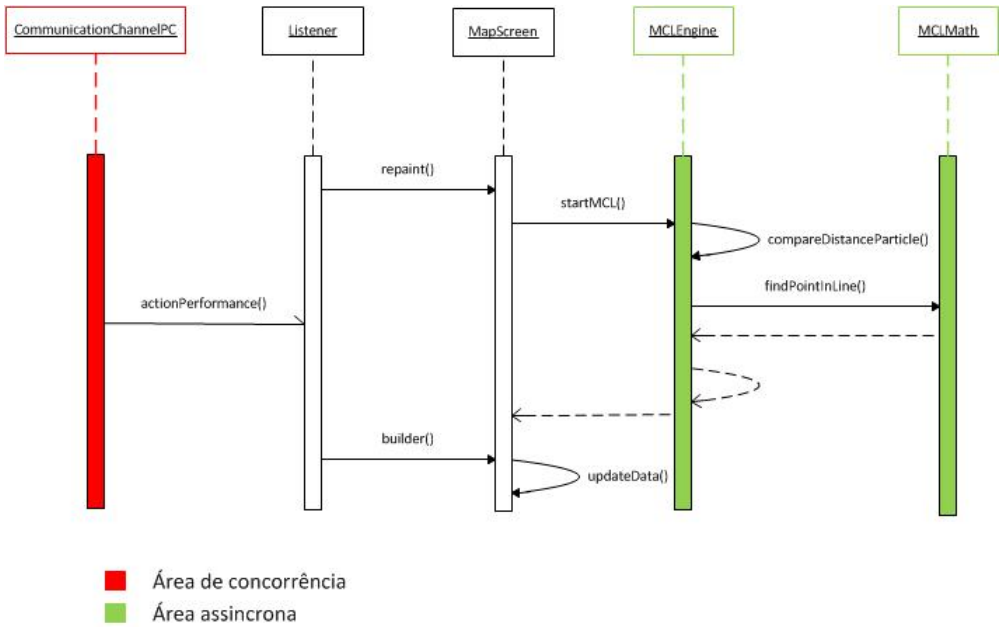

Figura 18: Diagrama de sequência do server

O diagrama de sequência apresentado pela figura 18 ilustra o caso mais complexo do módulo do servidor. O serviço de comunicação do módulo do servidor fica na escuta por dados enviados do NXT, após recebimento desses dados e adição na fila o Listener correspondente ao dado é disparado (pelo *MCLListener*) notificando o *MapScreen* e este, por sua vez, invoca o *MCLEngine*, que se responsabiliza por processar o algoritmo de localização que necessita da classe *MCLMath* para efetuar os cálculos de geometria. Lembrando que a área sinalizada em vermelho demonstra a região de concorrência e, a área em verde, assincronismo.

4.2.3 Componente Comunicação

O módulo de comunicação é compartilhado entre os outros dois módulos descritos anteriormente. Um projeto à parte, totalmente distinto de ambos os módulos, mas fundamental aos

componentes móvel e Server. É uma camada que gerencia toda a transferência de dados entre os módulos através da interface Bluetooth, que fornece uma maneira de se comunicar entre dispositivos por meios de frequências de 2.45 GHz, que não atinge grande alcance. Projetado como um Facede, que é uma interface que disponibiliza funções de uma forma simplista, esses componentes tem como responsabilidade abrir o canal de comunicação entre os dispositivos (móvel e fixo) permitindo a transmissão dos dados entre eles.

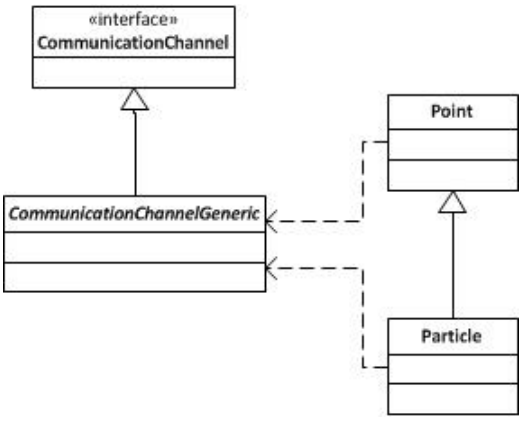

Figura 19: Diagrama de classe do módulo de comunicação

CommunicationChannel é a interface de comunicação entre os módulos; ela garante o contrato e a troca de dados, que é implementada pela *CommunicationChannelGeneric*, é uma classe abstratra que auxilia no gerenciamento das filas do robô e do Server. A *Point* é uma representação dos dados colhidos pelo NXT contendo as coordenadas cartesianas; a *Particle* classe, filha da *Point*, contém algumas informações a mais, como a orientação e ponto. Esta classe é utilizada na representação das partículas. Ambas as classes, *Point* e *Particle*, são os dados que trafegam entre os módulo móvel e Server.

4.3 Implementação do algoritmo de leitura e interpretação do ambiente

Todo o sucesso do algoritmo depende diretamente da leitura e interpretação das informações colhidas pelos sensores no ambiente, pois quanto mais precisa e pura for a informação melhor será o resultado produzido pela torre de controle. Os algoritmos de comportamentos, localização e mapeamento são alimentados por este pacote. Toda a leitura e interpretação do ambiente são feita através de dois sensores (Sonar e Toque), que são descritos a seguir. Mesmo não sendo um sensor, a odometria trabalha como coletor de dados, fornecendo informações fundamentais para o mapeamento, como as coordenadas cartesianas. Na sinalização de obstáculos, por um dos sensores (Sonar ou Toque), uma notificação de dados é disparada para o servidor, enviando dados cartesianos da posição do robô juntamente com a orientação de visualização e,

a distância em que o robô está do obstáculo.

Figura 20: Sensores Ultrasonic e Touch (LEGO, 2012)

4.3.1 Sonar

"A detecção dos obstáculos através do sonar é realizada utilizando-se o método das regiões de profundidade constante. Este método baseia-se no time-of-flight dos pulsos emitidos pelo sonar. Mais especificamente, define-se como time-of-flight o tempo entre a transmissão de um pulso e a recepção do seu eco refletido pelo alvo."(OTTONI, 2003)

A API do leJOS oferece uma classe (**lejos.nxt.UltrasonicSensor**) de controle para este sensor, tendo dois modos de trabalho para o contínuo (default), que envia pings de frequência sonora em curtíssimos intervalos de tempo, mantendo as informações de distância sempre atualizadas. A unidade de medida utilizada pelo sensor é em cm (centímetros), sendo o alcance máximo de 170 cm; em distâncias superiores a essa é retornado 255 o que significa que não há retorno da onda para o sonar.

4.3.2 Toque

O dispositivo de sensoriamento de toque entre em contato com um corpo fazendo pressão mecânica sobre ele, o que desencadeia pulsos elétricos, sinalizando a sua colisão. A classe **lejos.nxt.TouchSensor** é a que auxilia no controle do sensor com o leJOS. O seu uso é extremamente simples, sendo a verificação da pressão do sensor.

4.3.3 Odometria

"Odometria é o método de navegação mais utilizado para o posicionamento de um robô móvel, isso se deve ao fato de ser um modelo simples de localização que envolve apenas parâmetros internos do robô, e não depende da obtenção de informações do ambiente ao seu redor. Sabe-se que a odometria fornece uma boa precisão de curto prazo. É barata e permite taxas de amostragem muito altas. Um sistema de odometria consegue, com o auxílio de encoders, determinar a rotação dos eixos dos atuadores responsáveis pela movimentação do robô em um determinado intervalo de tempo, e integrando esses valores consegue calcular qual o deslocamento realizado nesse intervalo."(BOAS, 2011)

É comum a utilização de encoders acoplados aos eixos de robôs móveis conhecida como odometria. O seu conceito básico é a integração dos dados de movimento em um determinado tempo, ocasionando imprecisão. Essa técnica usa da rotação das rodas para calcular uma estimativa da posição do robô.

4.4 Implementação do algoritmo de navegação e mapeamento

Mapa é uma representação bidimensional visual de uma determinada região. A utilização de mapas no contexto robótico é fundamental para que o androide consiga gerar um modelo do ambiente em que está inserido, e utilizar dessa informação para cumprir sua missão. Para a construção do mesmo são necessários dispositivos que capturem informações do ambiente, feitos através de sensores que lidam com a luz, como a câmera ou infravermelho. Ao se movimentar pelo ambiente, o robô interpreta as estruturas e reage de acordo com o comportamento programado (BIGHETI, 2011). Para o robô desse projeto utiliza-se a odometria, que é um método muito simples de implantar em robôs móveis. Através dos movimentos das rodas, a interação da informação é incrementada no decorrer do tempo e do movimento, o que abre uma grande porte de erros, causando imprecisão na estimativa de localização do robô. Inicialmente não existe uma representação do ambiente em que o robô está inserido. A construção do mapa se baseia na odometria e na identificação de obstáculos através dos sensores acoplados ao robô. Apesar de ter um custo computacional maior, a construção do mapa em tempo de navegação é favorável a uma constante atualização. O gerenciamento da odometria é administrado pela API do Lejos através das classes **lejos.robotics.localization.OdometryPoseProvider** e **lejos.robotics.navigation.Navigator** que fornecem as coordenadas cartesianas do robô. Contudo, o mapa é uma representação cartesiana bidimensional do ambiente. Por movimentar-se em um ambiente desconhecido, foi necessário implementar comportamentos em que os mesmos

evitassem que o robô colidise com corpos presentes no mesmo.

A tabela 3 lista os comportamentos desenvolvidos e seus objetivos. Todas as classes são do pacote *org.reaction.rlm.nxt.navigator.behavior* e todas devem implementar a interface *lejos.robotics.subsumption.Behavior*

Comportamento (Behavior)	Descrição.
CollisionBehavior	Acionado quando o sensor de Touch é pressionado, interrompendo a movimentação do robô e desviando do obstáculo.
MCLBehavior	Coleta os dados necessários para o cálculo do algoritmo de localização.
NearbyObstacleBehavior	Ativado na identificação de um obstáculo a 25cm.
ScannerBehavior	Utilizado para scanner de obstáculos com o algoritmo de localização ativo.
WalkBehavior	Só desativa com o acionamento dos outros comportamentos, este é o que da movimento linear ao robô.

Tabela 3: Comportamentos

Os dados de navegação são colhidos do ambiente, interpretados pelos comportamentos e classificados antes de serem enviados para o servidor que utiliza os dados para localização e mapeamento. Por sua vez, o robô tem um controle dos últimos dados enviados a fim de otimizar a navegação.

4.4.1 Matriz de obstáculos

A matriz de obstáculos é uma representação dos elementos inseridos no ambiente. Cada célula da matriz pode representar uma área do ambiente em que pode existir ou não um obstáculo. Essa matriz apresenta características especiais das convencionais, pois nela existe uma sobreposição de eixos cartesianos, sendo que a origem se localiza no centro da matriz permitindo valores positivos e negativos conforme demonstrado na figura 21.

Com a identificação dos obstáculos, calcula-se em qual célula da matriz deve-se reproduzir, como por exemplo, um obstáculo nas coordenadas (6, -4). A leitura dos dados é feito simultaneamente com o encoder e o sonar, de tal maneira que possa ser convertida em coordenadas que irão representar o obstáculo na matriz. O conjunto da identificação de vários obstáculos converge na representação de um mapa, como ilustrado na figura 21.

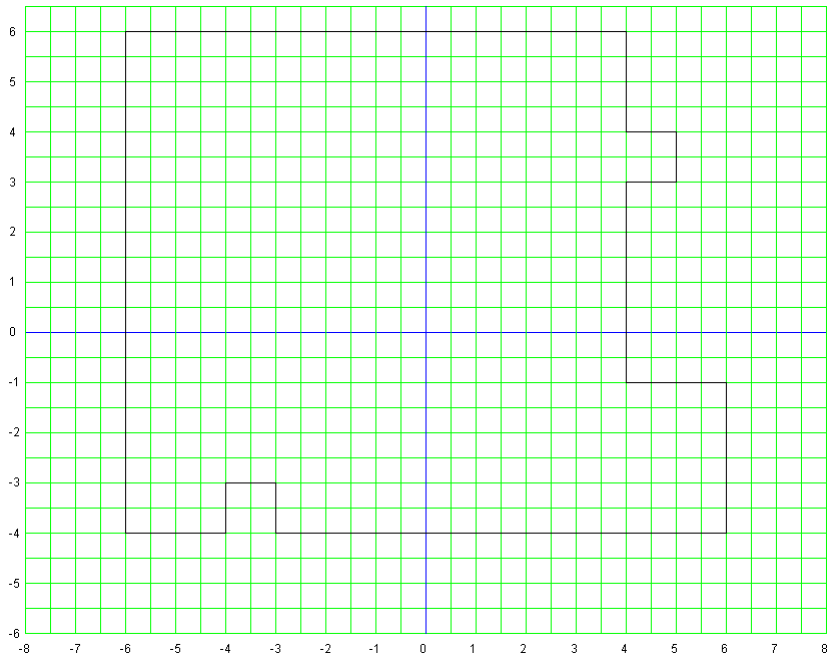

Figura 21: Matriz origem

4.5 Implementação do algoritmo de localização

O algoritmo de localização tem como responsabilidade determinar a posição do robô no ambiente, através de informações fornecidas pelos sensores acoplados no robô. A localização pode utilizar dos conhecimentos do mapa para calcular a posição. Existem metodologias que solucionam os problemas da localização e do mapeamento popularmente conhecidos como *SLAM (Simultaneous Localization And Mapping)*, em que a localização é feita simultaneamente com a construção do mapa, tornando-o mais fiel à realidade. Qualquer tarefa de navegação necessita determinar sua posição; essa informação é tão importante para a navegação quanto para a construção do mapa. Para obter precisão na posição do robô é preciso associar as informações de outros sensores com a odometria. Atualmente existem diversas técnicas para estimar a localização do veículo, tendo como foco o uso de métodos estatísticos como apresentado por (DELLAERT et al., 1999).

4.5.1 Localização e o método de Monte Carlo

O método de Localização Monte Carlo representa a probabilidade de envolvimento pela manutenção de um conjunto de amostras que são jogadas aleatoriamente. Usando uma representação dessas amostragens obtém-se um método de localização que pode representar distribuições arbitrárias. Apresenta resultados que são capazes e eficientes para localizar um robô móvel sem o conhecimento do seu local de partida. A implementação do algoritmo para este

trabalho é baseado nas informações do sonar. Utilizando o beneficio de movimentação horizontal, é possível utilizar como **scanner** e, rotacionando 360 graus, determinar as distâncias encontradas na região de cobertura. O processo integral do scanner é realizado por áreas menores de cobertura, a cada α graus é feita uma leitura da distância que o robô está do obstáculo, como demonstrado na figura 22.

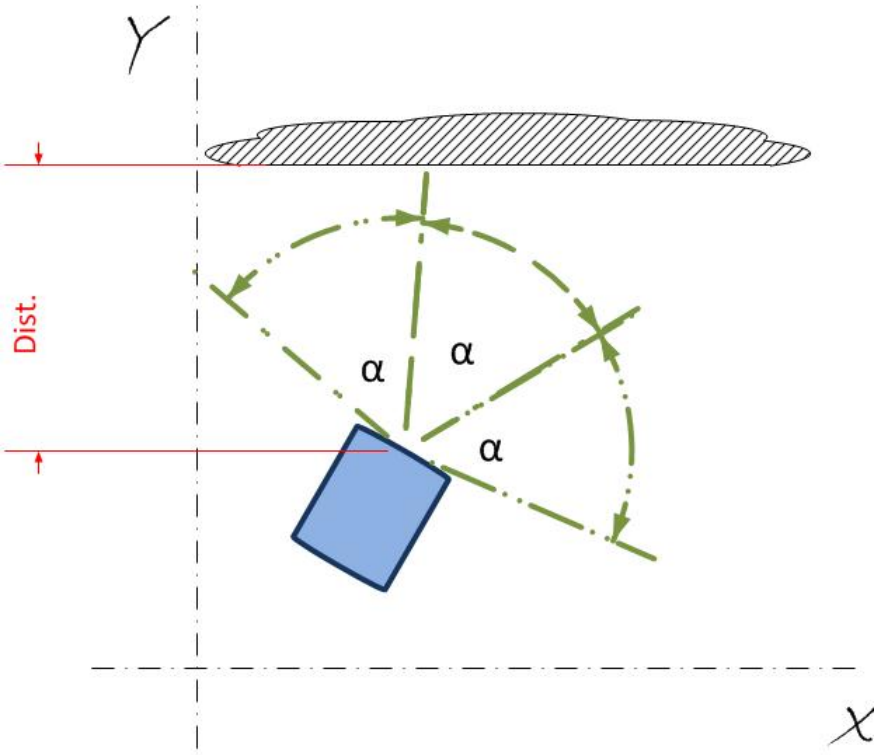

Figura 22: Scanner Sonar

A ideia inicial é calcular uma distribuição de pontos de probabilidades sobre todos os possíveis locais do ambiente. Esse conjunto de pontos p que representam uma posição no espaço onde x e y são coordenadas cartesianas e β representa a sua orientação. A distribuição dessas partículas expressa uma crença subjetiva de onde o robô pode estar logo, os valores (x, y e β) são gerados aleatoriamente conforme ilustrado na figura 23. Os procedimentos realizados no scanner são realizados com o robô é virtualizado para cada uma das partículas, gerando assim um conjunto de distâncias que cada uma delas está do obstáculo. Inicialmente são gerados N amostras distribuídas no plano e atribuído um fator de importância (*peso*) uniforme para cada uma das amostras.

As estimativas do estado das células são obtidas por meio da interpretação das leituras de distâncias usando modelos probabilísticos dos sensores. Para interpretar os dados sensoriais, é utilizado o cálculo de interseção entre segmentos de reta, considerando a distância máxima de alcance do sensor, obtendo assim, a distância entre o ponto de interseção com a partícula. Feitas

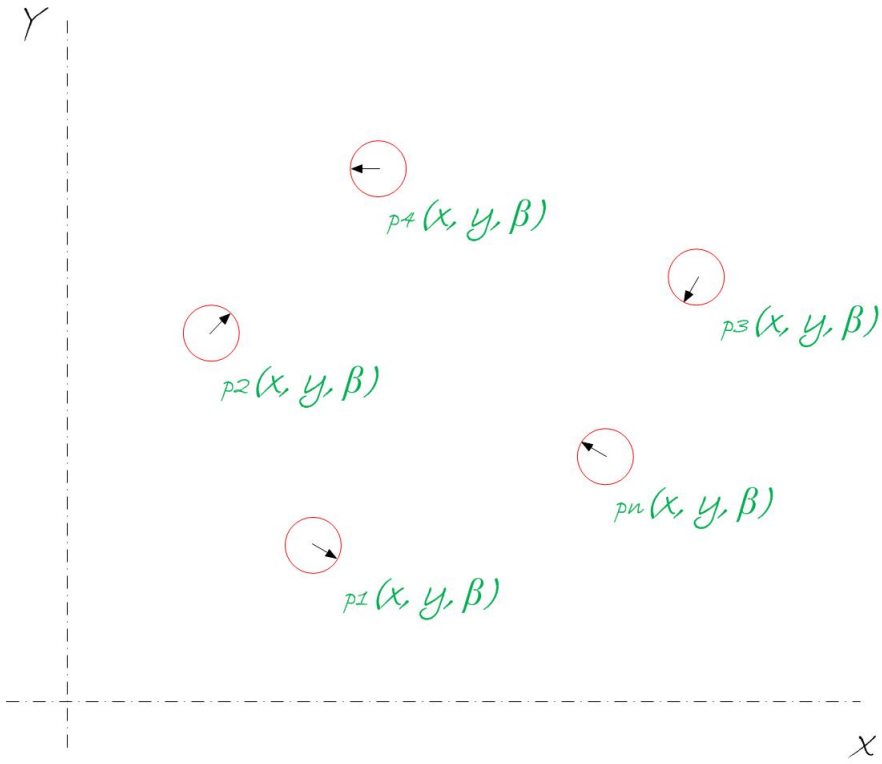

Figura 23: Partículas

as comparações de distâncias, calcula-se para cada partícula o fator probabilístico, aumentando ou diminuído a hipótese de localização do robô. O processo se repete a cada movimentação física do robô, a distância percorrida é replicada para cada uma das partículas e, executando todo o método probabilístico, refinando a hipótese de cada uma das partículas. A cada *n* ciclos, as partículas com menor crença são removidas, dando lugar a novas amostras geradas randomicamente. Todos os pesos das novas e das velhas partículas são uniformizados e o fluxo básico inicial é repetido. Desta forma, obtém-se uma concentração de amostras em algumas regiões próximas à localização real do robô.

5 EXECUÇÃO E ANÁLISE DO ALGORITMO

Neste capítulo são apresentados os resultados obtidos com a implementação da proposta descrita neste trabalho, portanto os experimentos práticos foram realizados com dados reais colhidos pelos sensores do robô. De modo geral, as análises foram feitas em duas etapas:

Teste de simulação: o algoritmo foi testado em uma plataforma virtual a fim de garantir o pleno funcionamento do mesmo, podendo diversificar os parâmetros de entrada.

Teste com robô real: o algoritmo desenvolvido foi implantando em um robô real a fim de analisar seu comportamento e aplicabilidade em ambientes reais.

Todo o algoritmo está disponibilizado para download pelo link *http://code.google.com/p/rlm/*(Google Code RLM).

5.1 Custo de pesquisa

O algoritmo foi desenvolvido utilizando ferramentas open-source disponível na web, para a diminuição do custo de pesquisa. Para tanto, foram utilizadas as seguintes ferramentas:

Ambiente de desenvolvimento IDE - Eclipse JUNO SDK 4.2.0;

Ferramentas de apoio - leJOS Development Tools 0.9.0.201202141552;

Linguagens de desenvolvimento - Java SE JDK 1.6;

Framework - leJOS NXJ 0.9.1.

Para a construção física do robô, foi necessária a aquisição do kit NXT 8547 da coleção Mindstorm fabricada pela LEGO, que gira em torno de 1.600,00 reais. Porém este gasto foi atribuído ao curso de Engenharia de Computação, que forneceu e disponibilizou o equipamento para a produção desse trabalho.

5.2 Teste de simulação

O ambiente de teste, em sua versão gráfica, permite que a análise de resultados seja mais fácil de ser interpretada. Para este caso de teste, é considerado o mapeamento do ambiente

conhecido. O foco deste teste é garantir a finalidade, qualidade e a execução do algoritmo. O algoritmo desenvolvido fornece uma interface gráfica de configuração simples e de saída gráfica adequada para a análise. O algoritmo apresentou resultados satisfatórios. Neste cenário a solução lida com 800 partículas e o resultado do refinamento de cada interação é dada em milésimos de segundos, atendendo os requisitos não funcionais de tempo real.

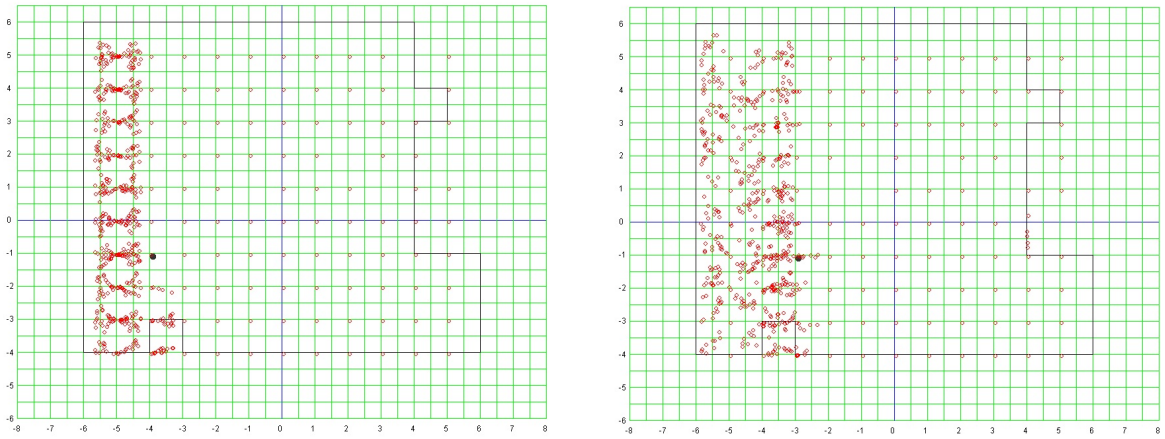

Figura 24: Teste de simulação - passo 1 e 2

A figura 24 demonstra o início da execução, onde as partículas (vermelhas, que representa a estimativa de localização) estão dispersas pelo ambiente no passo em que o robô(representada por um circulo preto) se movimentas as partículas se deslocam na mesma proporção. Ainda na figura 24 as partículas se dispersão espacialmente em orientações distintas.

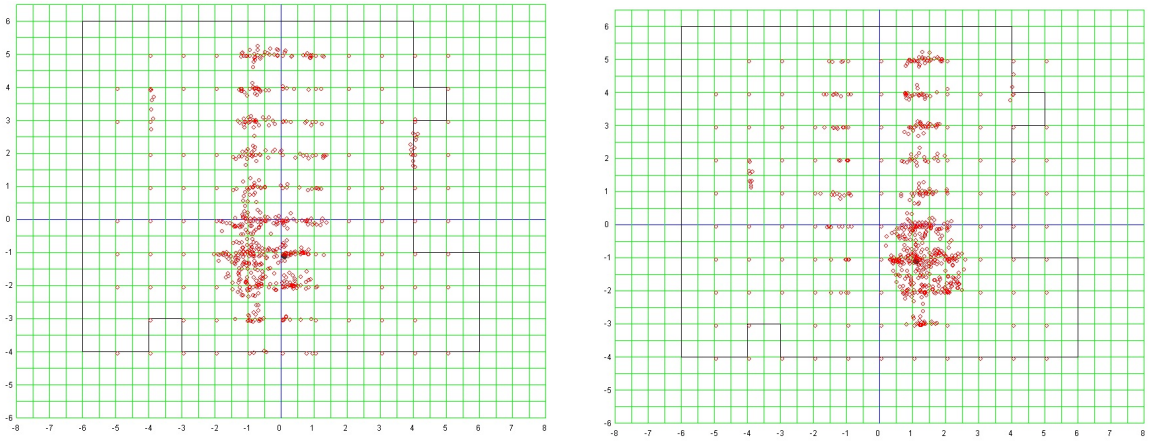

Figura 25: Teste de simulação - passo 3 e 4

Entre as figuras 24 e a 25 observa-se a movimentação do robô (círculo preto) e das partículas (círculo vermelho). Com o deslocamento das partículas a densidade aumenta próxima ao robô. O algoritmo atende, portanto, às propriedades descritas por (FOX et al., 1999) *"Por amostragem na probabilidade proporciona, MCL foca seus recursos computacionais em regiões com alta*

probabilidade, onde as coisas realmente importam.", lembrando que a qualidade dos resultados aumenta ao longo do tempo.

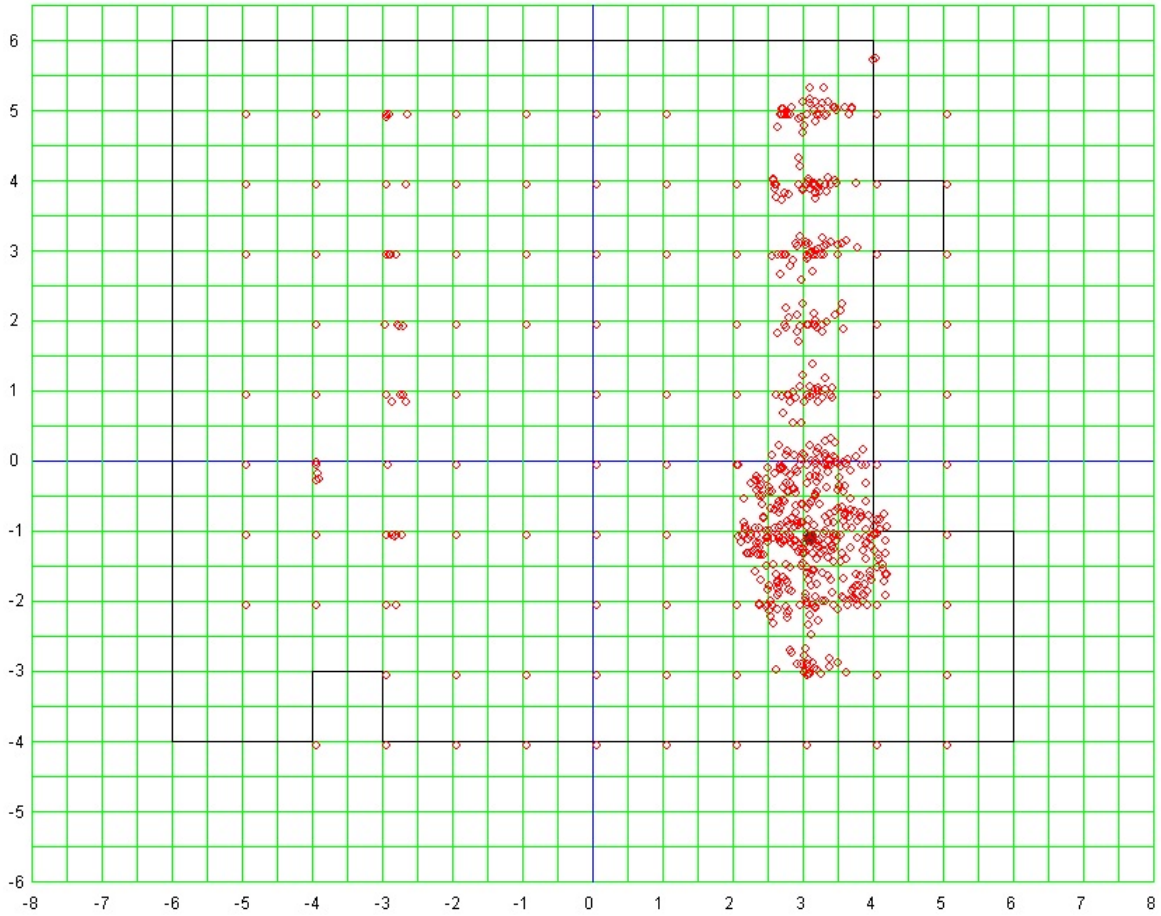

Figura 26: Teste de simulação - passo 5

O algoritmo obtido mostrou-se adequado ao objetivo de localizar o robô através de informações colhidas do ambiente, a um custo computacional baixo e de rápida resposta. O sistema apresenta instabilidades quando o robô passa muito próximo a um obstáculo.

5.3 Teste com robô real

5.3.1 Ambiente de teste

Assim como no teste de simulação, a interface gráfica faz parte do teste a fim de auxiliar na análise dos resultados com a diferença de dados reais. O local de teste influencia diretamente nos resultados. Visando validar a metodologia proposta, foi construído um ambiente de teste que se caracteriza por ser aberto e não dinâmico, podendo assim colher resultados claros.

5.3.2 Resultados

Os experimentos práticos realizados em ambiente real utilizam de dados coletados do mesmo por sensores (sonar e odometria) e processados em tempo de execução. Para a simulação foi utilizado o robô construído; navegando por todo o ambiente sem sinalização artificial, e a utilização do algoritmo de localização desenvolvido neste trabalho. O mapa do ambiente é dado conhecido pelo robô podendo avaliar o algoritmo localização. Contudo, os resultados obtidos foram bastante satisfatórios, uma vez que o robô adquiriu o objetivo de localizar-se no ambiente.

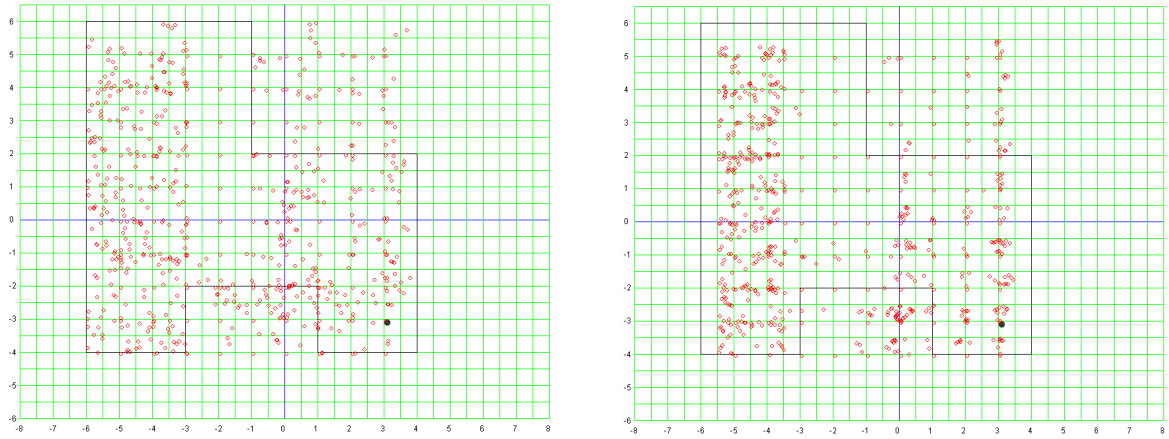

Figura 27: Teste com robô real - passo 1 e 2

Na figura 27 mostra o início da navegação e pode-se ver o funcionamento do algoritmo. Observando a referida figura, pode-se ver a movimentação das partículas vermelhas ainda dispersas no ambiente. Porém existe uma ligeira concentração delas entre os passos 1 e 2.

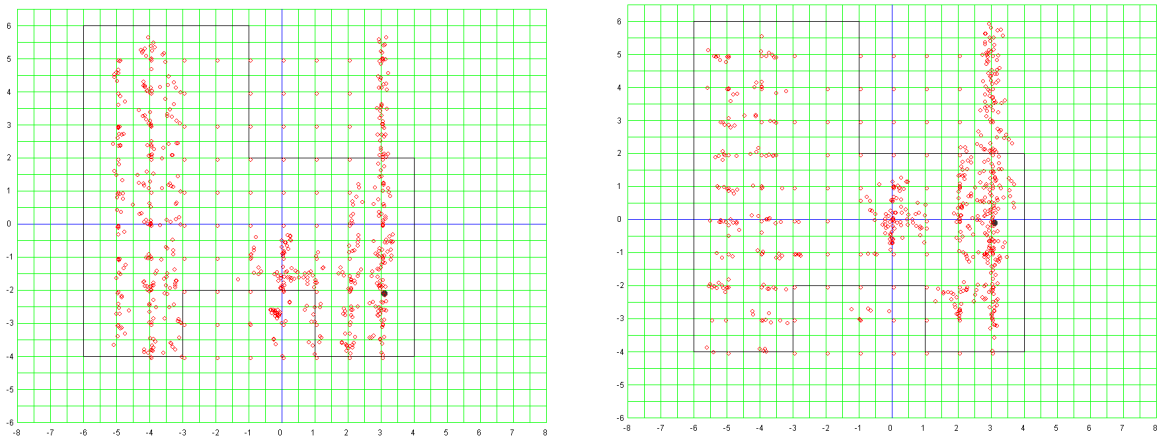

Figura 28: Teste com robô real - passo 3 e 4

Através da figura 28 é possível observar a utilização do algoritmo de localização com as alterações de valor das variáveis de navegação. Vale salientar que quando o robô efetua rota-

ções este colabora para o acumulo de erros na odometria, como o observado no decorrer dos testes, porém não o algoritmo de localização que se mantêm integra. Entre os passos 3 e 4, o concentração de partículas é mais clara uma vez em que a movimentação física do robô é maior.

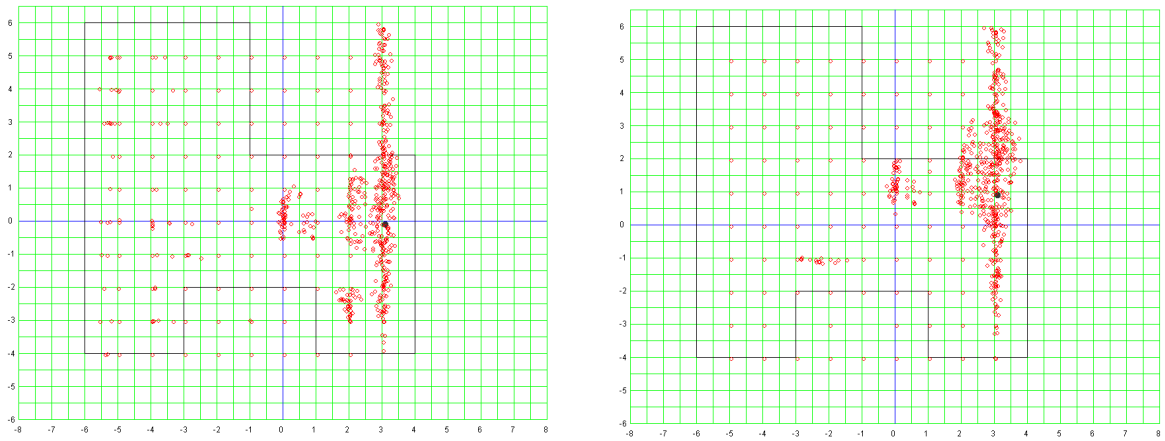

Figura 29: Teste com robô real - passo 5 e 6

Na figura 29 é possível notar uma densa concentração das partículas que representa um resultado significativo, porém ainda existe certa dispersão das mesmas enquanto no passo 6 era esperado um acumulo de partículas maior do que o apresentado.

6 CONCLUSÃO

Este trabalho apresentou uma implementação do algoritmo de localização para robôs móveis utilizando o Método de Monte Carlo, que se baseia em amostras para a localização do robô móvel. Inicialmente foi levada em consideração a utilização de imagens para auxiliar no algoritmo de localização, usando os conceitos oferecidos pelo *landmark*, o que agregaria grande valor ao trabalho, porém o tempo foi um fator decisivo para a sua eliminação. Contudo, outro desafio surgiu ao desenvolver o algoritmo de localização, utilizando somente um sensor sonar.

A primeira versão foi apresentada na 13º edição da feira de cursos da UNIARA, a FEC, entre os dias 30 e 31 de Agosto de 2012, que contemplava somente o algoritmo de mapeamento de ambiente. Esta implementação foi fundamental, pois se pôde explorar e conhecer os recursos oferecidos pela a API leJOS antes de desenvolver e aplicar o algoritmo de localização no robô real. Posteriormente, foi desenvolvido o algoritmo efetivo, fazendo-se uso dos mecanismos de navegação e coleta de dados, para processar a localização.

Tantos os resultados da simulação quanto reais, pôde comprovar a eficácia do algoritmo desenvolvido. Considerando que a proposta desenvolvida é suficientemente genérica que possa ser utilizada com outras soluções de navegação e mapeamento.

A ultima versão gerada para a conclusão e apresentação deste trabalho, foi apresentada no **12º CONIC-SEMESP** de 2012 (Congresso Nacional de Iniciação Científica,) entre os dias 30 de novembro e 1 de dezembro de 2012.

Como trabalhos futuros e seguindo a metodologia, sugere-se a que os próximos trabalhos se concentrem no aprimoramento do algoritmo para a utilização em ambientes dinâmicos, no auxilio à navegação e mapeamento dinâmico.

REFERÊNCIAS BIBLIOGRÁFICAS

ARKIN, R. C. **Behavior-Based Robotics**. 1. ed. Cambridge, Massachusetts: The MIT Press, 1998.

BIGHETI, J. N. NavegaÇÃo de robÔs em ambientes internos usando slam. In: **Tese De Mestrado pela UNESP-BAURU**. [S.l.: s.n.], 2011.

BOAS, E. R. V. Mapeamento e localização simultânea de ambientes dinâmicos aplicados na navegação de veículo autônomo inteligente. In: **PROGRAMA DE POS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA - UNIFEI**. [S.l.: s.n.], 2011.

CRACKED. Dezembro 2010. `http://www.cracked.com/blog/20-japanese-robots-probably-intent-on-murdering-you/`.

DELLAERT, F. et al. Monte carlo localization for mobile robots. In: **IEEE International Conference on Robotics and Automation (ICRA99)**. [S.l.: s.n.], 1999.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping. **IEEE/ASME - International Conference on Advanced Intelligent Mechatronic**, n. 1-1, p. 510–515, 2005.

FARINES JONI DA SILVA FRAGA, R. S. d. O. J.-M. **Sistemas de Tempo Real**. 1. ed. [S.l.: s.n.], 2000.

FOX, D. et al. Monte carlo localization: Efficient position estimation for mobile robots. In: **Proc. of the National Conference on Artificial Intelligence**. [S.l.: s.n.], 1999.

FOX, D.; THRUN, S.; BURGARD, W. **Probabilistic robotics**. [S.l.]: the MIT Press, 2005.

KINGSLEY-HUGHES, A. Dezembro 2010. `http://www.zdnet.com/blog/hardware/parrot-ar-drone-teardown/10575`.

LEGO. Julho 2012. `http://www.mindstorms.lego.com`.

LIU KUI YUAN, W. Z. Q. Y. J. Monte carlo multi-robot localization based on grid cells and characteristic particles. **IEEE Robotics and Automation Magazin**, n. 1-1, p. 96–108, 2006.

LORA ELDER M. HEMERLY, W. F. L. F. A. S. Estimação em tempo real de posição e orientação de robôs móveis utilizando sensores com diferentes taxas de amostragem. **CTA ITA IEEE**.

MIT. Agosto 2009. `http://info.abril.com.br/noticias/ciencia/peixes-robos-do-mit-tem-movimentos-reais-`

OTTONI, W. F. L. Guilherme de L. NavegaÇÃo de robÔs mÓveis em ambientes desconhecidos utilizando sonares de ultra-som. **Revista Controle Automação - Vol.14 no.4**, 2003.

PARKER, D. Julho 2012. `http://www.nxtprograms.com`.

SANTERIO, F. C. Controle baseado em comportamentos de robos moveis autonomos com sensores opticos e ultrassonicos. In: **Tese De Mestrado pela PUC-RIO**. [S.l.: s.n.], 2010.

SIRVA, C. R. Automação indutrial na indústria. **IEEE**, 2012.

SOLORZANO, J. Julho 2012. `http://lejos.sourceforge.net`.

SOUSA, A. J. M. de. Arquitecturas de sistemas robóticos e localização em tempo real através de visão. **Departamento de Engenharia Electrotécnica e de Computadores - Universidade do Porto**, 2003.

THRUN, S.; BüCKEN, A. Integrating grid-based and topological maps for mobile robot navigation. In: **Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence**. Portland, Oregon: [s.n.], 1996.

THRUN, S. et al. Robust monte carlo localization for mobile robots. **Artificial Intelligence**, v. 128, n. 1-2, p. 99–141, 2000.

WIJESOM, W. S. Toward multidimensional assignment data association in robot localization and mapping. **IEEE TRANSACTIONS ON ROBOTIC**, 2006.