Jales Monteiro Fernandes

Simulação para navegação autônoma de um robô móvel rastreador de objetos em ambientes não-planejados

Manaus
2014

Jales Monteiro Fernandes

Simulação para navegação autônoma de um robô móvel rastreador de objetos em ambientes não-planejados

Trabalho final de conclusão de curso elaborado durante a disciplina de trabalho de conclusão de curso II, submetido à Univesidade do Estado do Amazonas como parte do requisito para obtenção do título de Engenheiro Mecatrônico

Universidade do Estado do Amazonas
Escola Superior de Tecnologia
Graduação em Engenharia Mecatrônica

Orientador: Moisés Pereira Bastos. Me.

Manaus
2014

Jales Monteiro Fernandes

Simulação para navegação autônoma de um robô móvel rastreador de objetos em ambientes não-planejados

Trabalho final de conclusão de curso elaborado durante a disciplina de trabalho de conclusão de curso II, submetido à Univesidade do Estado do Amazonas como parte do requisito para obtenção do título de Engenheiro Mecatrônico

Trabalho aprovado. Manaus, 06 de dezembro de 2014:

Moisés Pereira Bastos. Me.
Orientador

Professor
Israel Francisco Benítez Pina. Dr

Professor
Luís Delfin Rolas Puron. Dr

Manaus
2014

Dedico este trabalho aos meus pais Josevam Fernandes e Wilma Monteiro pelo grande esforço e empenho que tiveram em suas vidas para dar a maior riqueza existente no mundo, a minha educação. Dedico também ao meu irmão Dr. James Monteiro Fernandes e a minha irmã, futura Engenheira, Wanessa Monteiro Fernandes, por sempre estarem presentes em minha vida e que juntos nunca deixamos de acreditar em uma vida melhor.

Agradecimentos

Agradeço primeiramente a Deus que me guiou e inspirou no decorrer dessa jornada.

A todos meus amigos de faculdade que durante a minha jornada neste período tiveram a confiança em trabalhar junto comigo e pude contar com suas ajudas que foram de fundamental importância para minha formação e que se um dia quiseram minha ajuda podem contar, pois estarei pronto a ajudar. Em especial aos meus amigos João Benevides, Maciel Castro, Ighor Cunha, João Guilherme, Rafael Toledo que serão amizades que levarei para resto da minha vida.

As meus familiares presentes e distantes, por ocasiões geográficas, que sempre depositaram muita confiança em mim e que sempre estavam dispostos a qualquer tipo de ajuda caso precisasse.

A meus amigos de infância, David Bezerra, Dalton Bezerra, Marcelo Gomes, Tiago Gomes, André Gomes, George Gomes, Angéliton Carlos muito obrigado pelo companheirismo tanto nas horas boas como nas difíceis.

Agradeço também a uma pessoa muito especial que me mostrou coisas boas na vida, que tem muita paciência em enfretar junto comigo as dificuldades de um relacionamento e esteve ao meu lado mesmo quando não acreditei que estaria, Daiane Bezerra.

A todos os docentes da Universidade do Estado do Amazonas que tiveram importâncias fundamentais no desenvolvimento do meu aprendizado, muitos que além do conteúdo específico davam incentivos e conselhos. Ao meu orientador Prof. Msc. Moisés Pereira Bastos que sempre esteve disposto a atender as dúvidas que sugiram durante a construção deste trabalho.

Enfim, a Universidade do Estado do Amazonas por me proporcionar a realização desse curso.

"O campo da derrota não está povoado de fracassos,
mas de homens que tombaram antes de vencer."
(Abraham Lincoln)

Resumo

Este trabalho trata da navegação de um robô móvel em ambiente no qual não se tem o conhecimento prévio do mapa, assim como a não preparação antecipada de alguns fatores tais como: taxa de luminosidade, superfície por onde o robô vai trafegar e tonalidade do ambiente em geral, o que caracteriza-se como sendo um ambiente não-planejado e fazendo uma abordagem para o problema de rastreamento de objetos. O algoritmo criado em C++ dará autonomia ao robô *Pioneer P3-AT* de forma que possa seguir objetos em movimento utilizando o *LASER SICK LMS200* para a percepção do ambiente. Como objetivo secundário será configurado um ambiente de simualação não somente para a validação da solução proposta para o problema de rastreamento, como também para inspeção futuras de soluções em robótica a fim de experimentar diferentes abordagens no campo do rastreamento, localização e mapeamento de robôs móveis.

Palavras-chaves: Engenharia Mecatrônica. Robótica. Robô Autônomo. Robô Pioneer P3-AT. Player/Stage.

Abstract

This work deals with the navigation of a mobile robot in an environment in which one has no prior knowledge of the map, as well as unanticipated preparation of some factors such as brightness rate, surface where the robot will to traffic and environmental tone in overall, which characterizes an unplanned environment by making an approach to the problem of object tracking. The algorithm created in C ++ will give autonomy to the robot Pioneer P3-AT so that it can follow moving objects using the SICK LMS200 *LASER* to the perception of the environment. As a secondary objective, a simulation environment will be set in the field of tracking, location and mapping of mobile robots.

Key-words: Mechatronics Engineering. Robotics. Autonomous robots. Pioneer P3-AT. LASER SICK. player/Stage.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas

D - distância entre o sensor *LASER* e os obstáculos
D'- Distância total percorrida pelo feixe de *LASER*
λ - comprimento de onda
α – constante relacionada a ângulos em graus
ηm – nanometro
m – metros
MHz - Mega Hertz
Δt - tempo de propagação do pulso entre o transmissor e o receptor
c - velocidade da luz no meio em que se propaga
m/s – Metros por segundo
Km - Quilômetros
SCARA - Selective Compliance Assembly Robot Arm
VANT's - Veículo aéreo não tripulado
SLAM - Localização e Mapeamento Simultâneos
LASER - Light Amplification by Stimulated Emission of Radiation
AGV's - Veículo Guiado Automatizado
LGV's - Veículo Guiado Por *LASER*
NASA - National Aeronautics and Space Administration
MATLAB - Matrix Laboratory

Sumário

1 Introdução

A robótica é uma área de conhecimentos que conjuga algumas disciplinas presentes da engenharia mecânica, com o estudo da estática e a dinâmica, de cálculo, com a descrição dos modelos dos robôs e do movimento no espaço, de engenharia eletrônica, com o projeto de sensores e interfaces para os robôs, de teoria de controle, com o projeto de algoritmos para que o robô realize os movimentos desejados, e, ainda, da computação, com a programação de todos os algoritmos desenvolvidos para que a tarefa designada para o robô possa ser realizada (FREIRE, 1997).

A robótica encontra-se dividida em duas grandes áreas: a robótica industrial, ou de manipulação e a robótica móvel.

Dentro da primeira área pode-se incluir a robótica a nível industrial, onde se destacam os robôs manipuladores, que foram os primeiros robôs a serem fabricados, ou alguns robôs usados na medicina para auxílio em cirurgias. Um exemplo é o robô manipulador do tipo SCARA, fabricado pela Toshiba.

Já na segunda área, a robótica móvel, incluem-se todos os robôs com capacidade de locomoção como, por exemplo, os robôs humanoides, os robôs usados em operações de inspeção, usados, por exemplo, para inspecionar zonas de dimensão reduzida , zonas perigosas, ou mesmo para exploração de outros planetas. Um exemplo de robô móvel *Patrobot (PATROLBOT, 2014)* , produzido pela *ActivMedia*. para fins domésticos

A aplicação prática de robôs móveis junto a diferentes atividades em nossa sociedade vem demonstrando o quão promissor é o futuro desta área (KLIPP, 2013). Por exemplo, seu uso em aplicações domésticas (aspiradores de pó e cortadores de grama robóticos), industriais (transporte automatizado e veículos de carga autônomos), urbanas (transporte público, cadeiras de rodas robotizadas), militares (sistemas de monitoramento aéreo remoto - VANT's, no transporte de suprimentos bélicos em zonas de guerra e no sistemas táticos de combate), de segurança na defesa civil e militar (controle e patrulhamento de ambientes, resgate e exploração em ambientes hostis), demonstram a grande gama de aplicações atuais dos robôs móveis e os interesses econômicos envolvidos em relação ao seu desenvolvimento e aplicação.

Dentre o meio acadêmico, é alvo de grandes pesquisas, pois se trata de uma área com muita capacidade para evolução, uma vez que a robótica móvel depende principalmente do avanço da tecnologia e desenvolvimento de algoritmos eficazes, possibilitando dessa forma que os estudantes desenvolvam projetos cada vez mais eficientes (MARQUES, 2014).

Outros pontos importantes quando se trata da robótica móvel estão relacionados à capacidade de perceber, modular, planejar e atuar para alcançar determinados objetos, sem a interferência ou mínima possível do fator humano, já que o robô pode se locomover em ambientes estruturados, pouco estruturados ou totalmente desconhecidos (PEDROSA,

2013) .

Um dos objetivos da robótica móvel é a criação de robôs autônomos. A robótica móvel enfatiza os problemas relacionados a locomoção dos robôs em ambientes complexos, que se modificam dinamicamente. Um robô autônomo deve ser capaz de navegar nestes ambientes sem que haja a intervenção humana, ou seja, deve adquirir e utilizar o conhecimento sobre o ambiente, ter a habilidade de detectar obstáculos e agir de tal forma que possa concluir a tarefa que lhe foi designada (HEINEN, 2002). O usuário apenas determina a tarefa a ser realizada e não a maneira como o robô deve agir para que a mesma seja concluída. Por apresentarem estas características, esses robôs são considerados dotados de certa "inteligência" e dependentes de técnicas de controle e de sensores. Essas interações que o robô deve ter com o ambiente de trabalho são conseguidas através dos vários sensores instalados a bordo do veículo.

1.1 Problematização

Um dos maiores problemas encontrados quando se trata de uma navegação autônoma está relacionado ao desconhecimento do ambiente, por parte do algoritmo embarcado em uma plataforma robótica que vai executar suas atividades, pois é muito complicado fazer esse reconhecimento do mapa local sem usar técnicas avançadas de Mapeamento e Localização Simultâneos - SLAM (COLARES; CHAIMOWICZ, 2012). Um fator complicante está na capacidade de discernimento entre o que é um obstáculo móvel e outro fixo quando não se dispõe de um ambiente conhecido. Outro fator de bastante dificuldade encontra-se no desenvolvimento desses sistemas, uma vez que para desenvolver é necessária a realização de vários testes. Para suprir essa necessidade existem as simulações virtuais, podendo economizar recursos financeiros, economia de tempo, danos às estruturas físicas do robô e o aperfeiçoamento das técnicas utilizadas.

1.2 Motivação

Com o surgimento de novas tecnologias torna necessária uma grande quantidade de testes quanto à eficiência, qualidade e viabilidade de um sistema em atingir resultados que satisfaçam o escopo estabelecido do projeto. Testes que realizados diretamente sobre o produto em questão podem ter um valores financeiros bastantes elevados e muitas das vezes perigosos.

Levando em consideração esses aspectos, uma maneira encontrada para reduzir esses riscos e custos é o uso das simulações virtuais. Elas podem apresentar algo muito próximo do real e se não chegam a eliminar testes reais, ao menos os reduzem em grande quantidade.

O desenvolvimento de uma navegação autônoma para um robô móvel, assim como o rastreamento de objetos em um ambiente não conhecido, logo são necessários vários testes possibilitando a correção de vários problemas que venham ocorrer, tais como, o erros de localização, que podem gerar trajetórias de colisão e podendo danificar a estrutura física do robô e seus acessórios, os erros também relacionados a uma má interpretação das informações provinda dos sensores e erros que podem colocar em risco as pessoas envolvidas na atividade.

Essas são questões que motivaram a busca por uma maneira de simular a navegação autônoma. Para isso, foi utilizado o ambiente de desenvolvimento *Player/Stage*, pois este vem sendo utilizado com sucesso e resultados satisfatórios ao redor do mundo (ROBOTS, 2014). Por isso como primeiro passo foi utilizado. Um segundo ponto importante na escolha dessa plataforma é o fato desse ambiente possuir código aberto, o que possibilita modificações necessárias para adaptação do projeto.

1.3 Justificativa

A utilização de robôs móveis, muitas vezes esbarra na dificuldade de desenvolver sistemas que tratam de navegação autônoma. Vários aspectos tem que ser levados em conta, dentre os de maior importância estão a identificação de obstáculos fixos e móveis e saber diferenciá-los, e que tipo inteligência irá ser empregada ao algoritmo para a percepção do ambiente na qual o robô fará sua navegação. Portanto essa necessidade da integração de sistemas buscando a navegação autônoma utilizando a simulação justifica o desenvolvimento do estudo desse tema.

1.4 Objetivo

O projeto tem a finalidade de desenvolver um algoritmo para que possa ser embarcado em um robô móvel, capaz de perseguir um objeto em movimento, levando em consideração o desvio de obstáculos fixos em um ambiente desconhecido pelo robô. Utilizar-se-á para essa captura o sensoriamento através do *LASER SICK LMS200* integrado a um robô *Pioneer P3-AT*.

1.4.1 Objetivo específicos

- Expandir os conhecimentos sobre robótica móvel autônoma;
- Testar, comparar e analisar as ferramentas presentes no simulador de robótica *Player/Stage;*
- Desenvolver competências para programação em softwares abertos de robótica móvel;

- Programar rotinas de detecção, tratamento e processamento das informações captadas no ambiente, pelos sensores do robô;
- Configurar um ambiente para a navegação de um robô autônomo.

1.5 Metodologia

A metodologia desenvolvida neste trabalho apresenta uma sequência de etapas para dispor as condições essenciais para o desenvolvimento do projeto.

Devido à dificuldade encontrada para adquirir um protótipo do robô que obedecesse às condições do trabalho proposto, foi necessário à utilização de um simulador que pudesse propor situações próximas às encontradas em um ambiente real.

Para que a simulação proposta fosse desenvolvida foram necessárias algumas etapas, tais como:

- Revisão teórica de cinemática e dinâmica de robôs móveis, esclarecendo as condições necessárias para que o robô proposto neste trabalho possa executar, obedecendo às leis físicas, assim como suas restrições na execução de tarefas;
- Estudo de algoritmos determinísticos para solução de problemas em robótica móvel, obedecendo às condições físicas reais, tais com o a velocidade a ser empregada, a localização do robô no ambiente em que ele será inserido;
- Estudar a ferramenta de simulação de robôs *Player/Stage* e as funções de interação com a interface do robô com os sensores, assim como a interpretação das informações obtidas externamente e enviar as instruções para que o robô execute a tarefa;
- Elaborar um algoritmo em C++ para autonomia do robô definido no qual a estrutura será constituída de uma maquina de estado e terá algumas configurações e inicializações de constantes preliminares para que o projeto proporcione um funcionamento dentro do esperado;
- Fazer a validação dos subprojetos em etapas tomando como base as etapas da maquina de estado, obedecendo as etapas dos estados configurados para que o robô tenha sua autonomia.

1.6 Apresentações dos capítulos

Este trabalho será apresentado em seis capítulos da seguinte forma:

O primeiro capítulo apresenta a proposta do projeto de conclusão de curso com: introdução ao tema, problemáticas, objetivos do trabalho, justificativa, motivação, a metodologia.

O segundo capítulo apresenta o estado da arte das principias áreas cientificas relacionada com este trabalho, também são tratados as aplicações, conceitos, definições de robôs e dos *LASERS*, os tipos de arquiteturas empregadas quando se trata da maneira como o controle inteligente se organiza.

O terceiro capítulo faz uma abordagem dos materias utilizados e métodos para a implementação do projeto, fazendo um estudo do *software Stage,* do robô *Pionner P3-AT* e no emprego do *LASER SICK LMS200*.

O quarto capítulo explana toda a execução da simulação da navegação autônoma desenvolvida pelo algoritmo, mostrando cada etapa da máquina de estado, destacando uma série de considerações e requisitos para execução dos objetivos.

No quinto capítulo é mostrado os resultados e as etapas de validação no simulador em cada etapa da máquina de estado .

No sexto capítulo apresenta-se a conclusão e trabalhos futuros.

2 Robótica móvel

2.1 Estado da arte

Neste capítulo será abordado o estado da arte das principais áreas cientificas relacionada a este trabalho. A primeira sessão descreve alguns robôs móveis, desde os primeiros desenvolvidos até os mais recentes, assim como seus centros de desenvolvimentos, campo de pesquisa e suas aplicações. Serão também abordados conceitos da robótica móvel que trata da sua classificação pela mobilidade, a modelagem cinemática, seus tipos de controle de arquitetura e conceito de *LASER*[1]

2.2 Rôbos móveis

A *Robotic Industries Association*[2] define por robô, um manipulador programável capaz de tomar decisões utilizando métodos de raciocínio lógico multifuncional e capaz de realizar uma variedade de tarefas, seja para mover materiais, ferramentas ou dispositivos específicos através de movimentos programados (NEHMZOW, 2009). No entanto, essa definição abrange toda a categoria de máquinas e considera portanto, um robô como qualquer equipamento capaz de ser programado.

A robótica industrial, cuja maioria dos robôs possui grandes dimensões e bases fixas são capazes de realizar tarefas envolvendo a manipulação de objetos em sua área de trabalho, esta é, todavia limitada a um espaço restrito. Além disso, robôs industriais são no geral desprovidos de inteligência de alto nível, considerando-os dessa forma máquinas-ferramentas (ROSáRIO, 2008) .

Por outro lado, a robótica móvel dispõe obviamente de maior versatilidade no que tange à mobilidade do robô. Atividades que exigem uma movimentação mais ampla podem ser feitas através de rodas, pernas ou hélices. Robôs móveis são capazes de sensoriar o ambiente, movimentar-se e tomar decisões por meios sistemas embarcados e plataformas responsáveis pela integração de *software* e *hardware*.

Com o desenvolvimento da tecnologia empregada em robôs móveis, suas utilizações vêm sendo cada vez mais requeridas em processos industriais, tecnológicos e pesquisa em geral(JUNG, 2005). Como exemplos famosos de robôs móveis, resultante de pesquisas e desenvolvimentos que vem ocorrendo nessa área, podemos citar os robôs de exploração espacial como o *Mars Pathfinder's Sojourner, Spirit* e o *Opportunity Rovers* (BAJRACHARYA, 2008), os de serviços domésticos como o *Roomba* e *Scooba (ROOMBA, 2014).*

[1] *Light Amplification by Stimulated Emission of Radiation*
[2] *Associação Indústrias robótica*

Podemos citar também os Veículos Aéreos Não Tripulados, como os VANT's brasileiros do projeto Arara (NERIS, 2001).

Desenvolvido no *Stanford Research Institute*[3] no ano de 1968, o robô *Shakey,* figura 1, era equipado por sensores de presença e câmera de visão dando a possibilidade de locomoção em superfícies planas e tinha como principal objetivo empurrar blocos. Era controlado por um computador de grande porte dotado de um raciocínio inteligente e seletivo. Este foi o primeiro robô móvel de que se tem registro.

Figura 1 – Shakey, o primeiro robô móvel no ano de 1968.

Fonte: (http://www.frc.ri.cmu.edu/~hpm/talks/Robot. Figures/fig.ch2/p027.html). Acesso em 13/04/2014.

Outro robô móvel reconhecido, nos registros, no ano de 1977 foi o Stanford Cart, ilustrado na figura 2, desenvolvido no *Stanford Artificial Intelligence Laboratory*[4]. Este robô utilizava um sistema de navegação baseado no "parar e seguir". A cada metro percorrido, o robô fazia uma parada e por meio de seus sensores era feito uma leitura do ambiente. Finalmente era realizado o planejamento da trajetória (VIEIRA, 2005).

[3] *Instituto de Pesquisa Stanford*
[4] *Laboratório de inteligência artificial de Stanford*

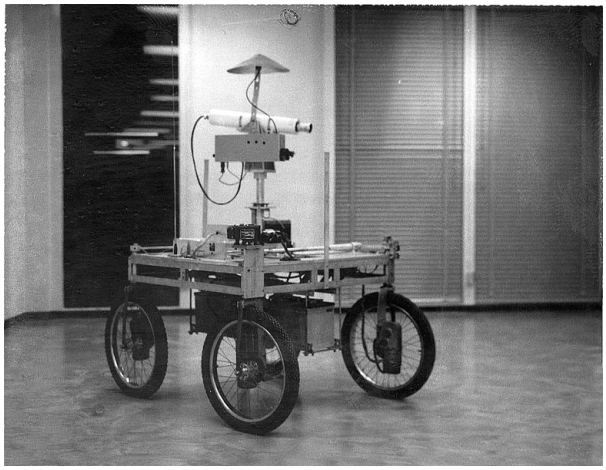

Figura 2 – Robô Stanford Cart, ano de 1970.

Fonte: (http://www.frc.ri.cmu.edu/~hpm/talks/Robot. Figures/fig.ch2/p027.html). Acesso em 25/06/2014

Na década de 80, vários foram os trabalhos desenvolvidos em todo o mundo nesse segmento da robótica móvel, começaram a aparecer projetos para aplicações como patrulhamento de ambientes, transporte de cargas, segurança, exploração, órgãos militares, governamentais ou para grandes empresas (EVERETT, 2014). Vale a pena ressaltar o robô *Hilar Laas*, figura 3, desenvolvido no *Laboratoire d'Architecture et d'Analyse de Systémes*[5] no ano de 1983, equipados com sensores ultrassônicos e telemetria *LASER*(VIEIRA, 2005).

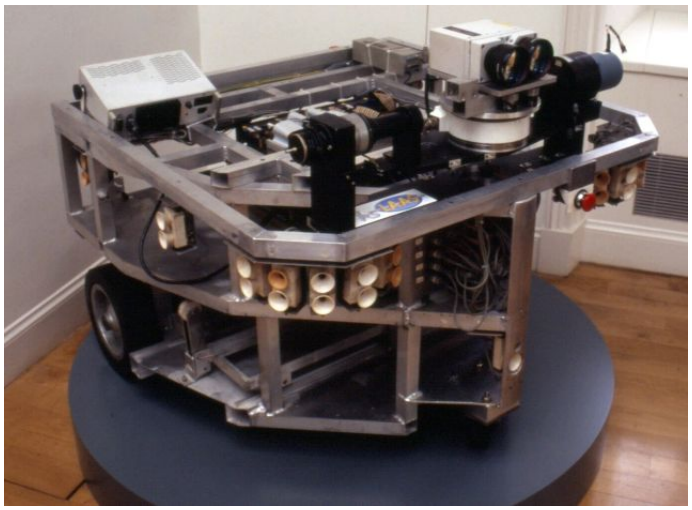

Figura 3 – Robô Hilare, no ano de 1983.

Fonte: (http://homepages.laas.fr/matthieu/robots/images/hilare.gif). Acesso em 25/06/2014.

Nos anos 90 continuou o grande avanço com o desenvolvimento de vários robôs realizando atividades em ambientes nocivos à atividade humana. Desde robôs com finalidade

[5] *Laboratório de Arquitetura e Análise de Sistemas*

de exploração em ambientes marítimos ou até mesmo realizando atividades em vulcões ativos, como é o caso do robô Dante II, figura 4, desenvolvido no *Field Robotics Center*[6], que foi responsável em fazer a exploração de vulcões no Alaska, utilizando a técnica de Rapel.

Figura 4 – Robô Dante II, no ano de 1994.

Fonte: (http://www.frc.ri.cmu.edu/projects/danteII). Acesso em 16/07/2014

Hoje é possível encontrar robôs fazendo a exploração de forma autônoma no espaço sideral, como é o caso no robô "Polaris", figura 5, em desenvolvimento desde 2012 pela *Astrobotic Technology*[7] , que será enviado em 2015 para uma missão na Lua com o objetivo de fazer a busca de regiões que possuam depósitos de água congelada. Portanto é possível encontrar robôs móveis nas mais diferentes formas, tamanhos e funcionalidades, o que lhes compete uma infinidade de aplicações, desde e as científicas, industriais e até mesmo educacionais e entretenimento.

[6] *Campo de Centro de Pesquisas em robótica - CMU*
[7] *Astrobotic Tecnologia*

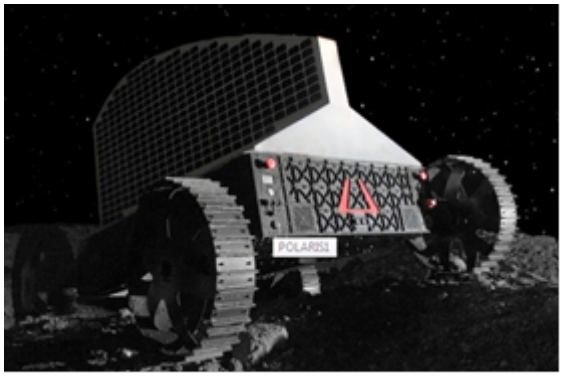

Figura 5 – Robô Polaris, 2012

Fonte: (http://www.frc.ri.cmu.edu/projects/danteII). Acesso em 16/07/2014

2.3 Aplicações dos rôbos móveis

Atualmente no mercado mundial existem cinco classificações para as funções de um robô móvel, que são as indústrias, de serviço, para pesquisa, de campo e para o entretenimento (HEINEN, 2002).

2.3.1 Aplicações industriais

São plataformas móveis, que normalmente executam atividades de transporte de peças ou equipamentos pesados. Esse tipo de robô normalmente tem pouca interação com os seres humanos, pois possuem uma autonomia muito grande (HEINEN, 2002). Normalmente a navegação utilizada para estes tipos de robô são através de seguidores de linhas, que por sua vez pode acarretar em uma mudança no *layout* da empresa a fim gerar um ambiente seguro para suas navegações . Um exemplo são os robôs dos tipos *Automated Guided Vehicle*[8] (AGV's) e *LASER Guided Vehicle*[9] (LGV's), figura 6 e figura 7, respectivamente.

[8] *Veículo Guiado Automatizado*
[9] *Veículo Guiado Por LASER*

Exemplo de um robô móvel do tipo LGV

Figura 6 – Exemplo de um robô móvel AGV

Fonte: (Felippe de Souza).

Figura 7 – Exemplo de um robô móvel do tipo LGV

Fonte: (Felippe de Souza).

2.3.2 Rôbos de serviços

A robótica de serviço tem seu papel fundamental devido a projeção no futuro, pois será uma área de grande pesquisa (HEINEN, 2002). Os serviços que essa tecnologia vem trazendo estão relacionados às tarefas comuns, mas de grande importância no dia-a-dia, como é o caso de transporte, manipulação, limpeza, vigilância e etc. podendo exercer suas atividades de forma autônoma sem a preocupação se o ambiente é estruturado (HEINEN, 2002).

Um bom exemplo desse tipo de robô é o "Chico Mendes", figura 8, desenvolvido pela empresa Petrobrás. Este robô faz parte de um projeto de monitoramento ambiental do gasoduto Coari-Manaus na região Norte do Brasil, percorrendo cerca de 700km às margens do Rio Solimões.

Figura 8 – Robô Chico Mendes

Fonte: (http://exame.abril.com.br/tecnologia/fascinante-passeio-pelo-mundo-dos-robos). Acesso em 01/08/2014

2.3.3 Robôs de campo

Esse tipo de robô está relacionado às atividades em superfícies não estruturadas, pouco conhecidas e ambientes nocivos aos seres humanos. Hoje esses tipos de robôs vêm realizados muitas atividades em exploração espaciais, exploração de minas, acidentes nucleares, navegação em autoestrada, entre outras diversas atividades.

Um bom exemplo é o robô Curiosity, figura 9, desenvolvido no laboratório do *California Institute of Technology*[10] em parceria com a NASA que é considerado em laboratório móvel em solo marciano. Seu pouso em Marte foi realizado dia 6 de agosto de 2012 e suas análises sobre a superfície da Marte já vem sendo estudadas.

[10] Institudo de tecnológia da California

Figura 9 – Robô Curiosity.

Fonte: (http://veja.abril.com.br/noticia/ciencia/robo-curiosity-encontra-agua-em-solo-marciano). Acesso: em 01/08/2014

2.3.4 Robô para pesquisa

Essa categoria está relacionada principalmente ao meio acadêmico, no setor de desenvolvimento e pesquisa de tecnologias. Muitas empresas produzem projetos e faz a comercialização de seus modelos, como é o caso do robô *Khepera*, figura 10. Trata-se de um robô de dimensões reduzidas e valor de mercado atrativo, fator que o torna muito popular nas universidades.

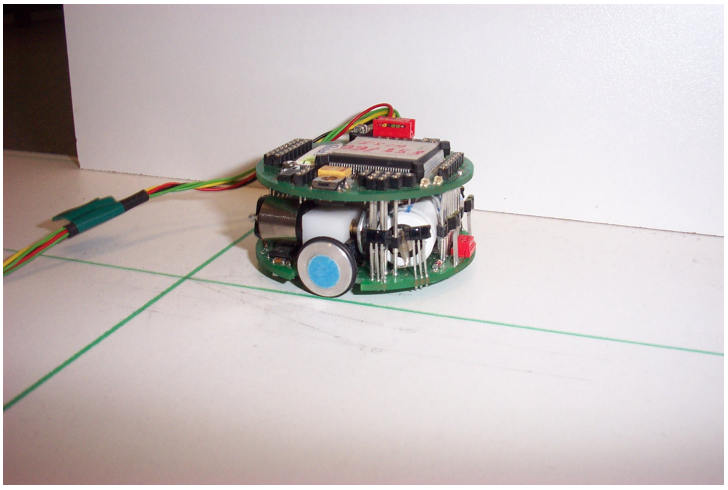

Figura 10 – Robô Khepera

Fonte: (http://www.k-team.com/gallery/index.html). Acesso: em 01/08/2014

2.3.5 Robô para o entretenimento

Esse tipo de robô é talvez o maior alvo de propagandas no segmento da robótica. Um exemplo é o *ASIMO*, figura 11, desenvolvido e projetado pela Honda.

Figura 11 – Robô móvel ASIMO.

Fonte:
(http://asimo.honda.com/ASIMO_DCTM/News/images/highres/ASIMO_Zurich___1_.jpg).
Acesso em 01/08/2014

2.4 Conceitos de definições

Os robôs móveis podem ser classificados em três grupos de acordo com a mobilidade que ele exerce (PIERI, 2002), como ilustra a figura 12.

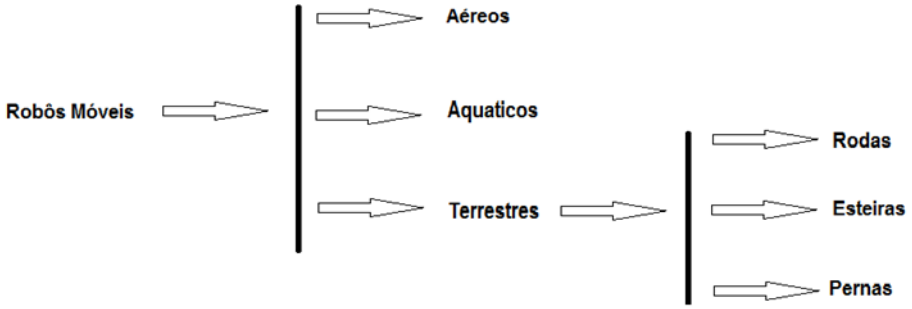

Figura 12 – Esquema de classificação de acordo com De Pieri.

Fonte: (Próprio autor).

Classificam-se robôs móveis aéreos como sendo aqueles que fazem sua trajetória no ar. São considerados os mais difíceis de controlar e os que exigem um grau maior de planejamento em virtude do valor dos equipamentos terem um custo bastante elevado, sem desconsiderar os riscos que podem trazer para as pessoas envolvidas(PIERI, 2002). O aeromodelismo é o exemplo mais recorrente.

Os robôs aquáticos podem ter suas atuações na superfície da água ou abaixo dela, controladas remotamente ou com autonomia própria. São equipados com sensores medidores de pressão e atuadores que podem ser propulsores ou balões de ar, criando dessa

forma uma interação com o ambiente aquático.

Robôs móveis com rodas - são assim chamados porque utilizam para sua locomoção uma ou mais rodas. Quando há a necessidade da utilização deste tipo de robô é levado pouca consideração em relação ao equilíbrio, uma vez que o robô está sempre em contado com o solo. Dessa forma, a maior preocupação envolve o controle de tração, estabilidade e manobras.

Robôs móveis com esteiras – são robôs que possuem uma maior manobrabilidade em terrenos irregulares, devido à maior fricção com solo. São geralmente mais robustos, dotados de sensores na parte superior, o que permite uma melhor navegação nesse tipo de terrenos. É muito utilizado para busca e resgate, desarmamento de bombas e etc.(BRäUNL, 2008).

Robôs móveis com pernas – são robôs dotados de pernas para efetuarem a locomoção, inspirados em animais, denominados assim de antropomórficos, caracterizados por pontos de fixação no chão. Não havendo a necessidade da fixação das pernas no solo esse tipo de robô mantem uma maior estabilidade em terrenos irregulares (SIEGWART, 2004).

2.4.1 Modelo do robô móvel

Os modelos cinemático e dinâmico para um robô que faz sua locomoção utilizando quatro rodas deslizantes e fazendo sua mudaça de orientação por aplicação de torques como valores distintos para cada motor, é baseado em (KOZLOWSKI; PAZDERSKI, 2004). Estes modelos são utilizados pelo método de planejamento de movimento.

2.4.1.1 Modelo Cinemático

A estrutura geometrica do robô é ilustrada na figura 13, representada no plano de coordenadas (X, Y) e tendo como referências inerciais X' e Y'. O sistema de coordenadas fixo no corpo tendo como origem o centro de gravidade do robô CG e a coordenada do centro de gravidade do robô no sistema de coordenadas inerciais é (x, y), e ψ é o ângulo entre o eixo X' e o eixo X. As coordenadas do centro instantâneo de rotação (CIR) são (x'_{CIR}, y'_{CIR}). O parâmetro a corresponde ao eixo Y', b a distância entre o centro de gravidade e o eixo de simetria paralelo ao eixo Y' da roda atuada traseira, c é a metade da distância entre as rodas ao longo do eixo Y' e r o raio das rodas.

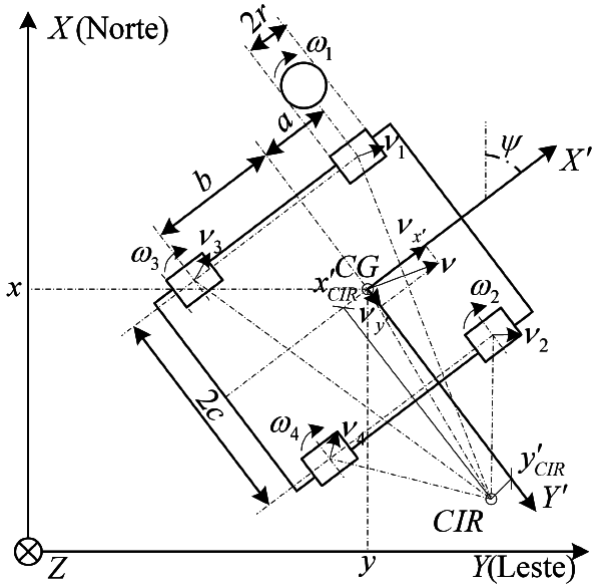

Figura 13 – Geometria do Robô Móvel

Fonte: (OLIVEIRA VAZ, 2011)

As velocidades absolutas ẋ, ẏ e ψ' no sistema de coordenadas inercial são dadas por:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \end{bmatrix} = \begin{bmatrix} cos\psi' & -sin\psi' & 0 \\ sin\psi' & cos\psi' & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \upsilon_{X'} \\ \upsilon_{Y'} \\ \omega \end{bmatrix} = \mathrm{R}\psi' \begin{bmatrix} \upsilon_{X'} \\ \upsilon_{Y'} \\ \omega \end{bmatrix}$$

Diferenciando a equação acima em relação ao tempo tem-se

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \psi'' \end{bmatrix} = \mathrm{R}_{(\psi)} \begin{bmatrix} \acute{\upsilon}_{x'} - \psi' \acute{\upsilon}_{y'} \\ \acute{\upsilon}_{y'} + \psi' \acute{\upsilon}_{x'} \\ \acute{\omega} \end{bmatrix} = \mathrm{R}_{(\psi)} \begin{bmatrix} a_{x'} \\ a_{y'} \\ \acute{\omega} \end{bmatrix}$$

Diferentemente da maioria dos robôs móveis, a velocidade lateral de um robô móvel com quatro rodas é geralmente diferente de zero, isso vem da estrutura mecânica dessa característica que faz o deslizamento lateral necessário para que o veículo mude de orientação, e quando a velocidade lateral $\upsilon_{y'} = 0$ não há deslocamento lateral. A velocidade angular ω e a velocidade lateral $\upsilon_{y'}$ ambas desaparecem quando o robô movimenta-se em linha reta, e o *CIR* vai para o infinito ao longo do eixo Y'. Já em uma trajetória curva o desloca x'_{cir} ultrapassar a base das rodas do robô, o veículo desliza drasticamente com perda de estabilidade de movimento. assim, para completar o modelo cinemático do robô de quatro rodas, a seguinte restrição não-holonômica foi introduzida (LUCA, 1999))

$$\upsilon_{y'} = x'_{cir}\psi, \text{ com } x'_{cir} \in \text{(-b, a)}.$$

No entanto, da figura 13 obtém-se

$$\upsilon_{y'} = \text{-sin}\psi\dot{x}\ \text{cos}\dot{y}\psi$$

seja $\boldsymbol{q} = \begin{bmatrix} x & y & \psi \end{bmatrix}^T$, o estado do robô, substituindo nas equações anteriores ficará da seguinte maneira:

$$\begin{bmatrix} -sin\psi & cos\psi & -x'_{cir} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \psi' \end{bmatrix} = \text{A(q)q'} = 0$$

A partir da figura 13, as velocidades q' podem ser expressas como:

$$\text{q'}= \text{S(q)}n$$

onde

$$\text{S}_{(q)} = \begin{bmatrix} cos\psi & -x'_{cir}sin\psi \\ sin\psi & x_{cir}cos\psi \\ 0 & 1 \end{bmatrix} n = \begin{bmatrix} \upsilon_{x'} \\ \omega \end{bmatrix}$$

$\text{S}_{(q)}$ é uma matriz de posto completo, cujas colunas estão no espeço nulo de $\text{A}_{(q)}$, ou seja

$$\text{S}^T{}_{(q)}\text{A}^T{}_{(q)}= 0$$

2.4.1.2 Modelo dinâmico

A figura 14 ilustra as forças de tração $\text{F}_{x'i}$que são sujeitas a forças resistivas longitudinais $\text{R}_{x'i}$. Um robô de quatro possui normalmente dois eixos que são responsáveis por acionar as rodas de um lado do robô. Então assume-se que a atuação das rodas é igual em cada lado, ou seja, $\text{F}_{x'1}= \text{F}_{x'3}$e $\text{F}_{x'4}$,diminuindo assim o deslisamento lateral. O momento resistivo $\text{M}_r$em volta do centro de massa, que se opõe ao momento M é induzido pelas forças $\text{R}_{y'i}$ e $\text{R}_{x'i}$.

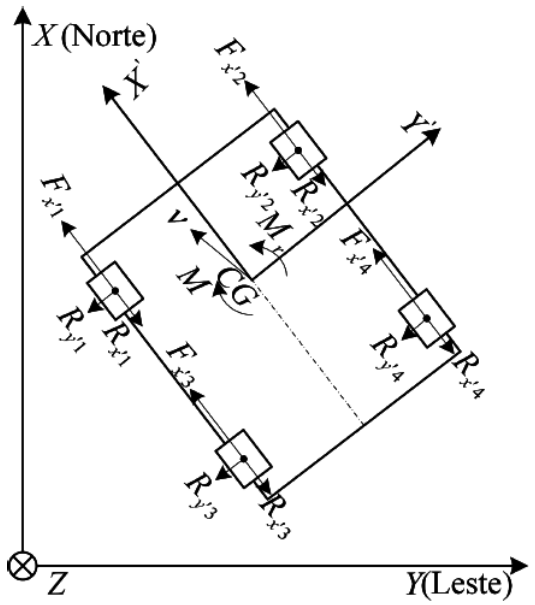

Figura 14 – Forças de tração e resistivas do robô móvel.

Fonte: (OLIVEIRA VAZ, 2011)

Para um robô, como ilustrado na figura 14, de massa m e de inércia I no centro de massa, as equações de movimento no sistema de coordenada do corpo são

$$ma_{x'} = 2F_{x'1} + 2F_{x'2} - R_{x'}$$

$$ma_{y'} = -R_{y'}$$

$$I\psi'' = 2c(F_{x'1} - F_{x'2}) - M_r$$

Para representar as forças longitudinais $\mathrm{R}_{x'}$as forças laterais $\mathrm{R}_{y'}$, e o momento resistivo M_r, deve-se considerar mg como a carga gravitacional, sendo m a massa do robô e g a aceleração da gravidade, que é dividida sobra as rodas do robô, e introduzir o modelo Coulomb de atrito para o contato das rodas para o contato das rodas com a superfície. Desse modo, desde que o robô tenha uma linha mediana longitudional de simetria, obtém-se

$$\mathrm{R}_{z'1} = R_{z'2} = \tfrac{b}{2(a+b)}mg;$$

$$\mathrm{R}_{z'3} = R_{z'4} = \tfrac{a}{2(a+b)}mg.$$

Em velocidades baixas, a carga lateral transferida devido as forças centrifugas sobre percusos pode ser desconsiderada. No caso de superfícies duras, pode-se assumir que o contato entre as rodas e o solo é retangular e que a carga vertival produz uma pressãp uniformemente distribuída. Assim $\mathrm{R}_{x'i} = fR_{z'i}sgn(\upsilon_{x'i})$ sendo que: f_ré o coeficiente resistivo de rolagem independente da velocidade e sgn(.) a função sinal. A força resistiva longitudinal total é

$$\mathrm{R}_{x'} = \sum_{i=1}^{4} = R_{x'i} = \frac{f_r}{2}(sgn(\upsilon_{x'2})),$$

Intriduzindo o coeficiente de atrito lateral u, temos que $\mathrm{R}_{y'i} = uR_{z'i}sgn(\upsilon_{y'i})$. Portanto a força lateral total é dada por

$$\mathrm{R}_{y'} = \sum_{i=1}^{4} R_{y'i} = u\frac{mg}{a+b}(bsgn(\upsilon_{y'1}) - asgn(\upsilon_{y'3})),$$

Enquanto o momento resistivo será

$$M_r = u\frac{abmg}{a+b}(sgn(\upsilon_{y'1} - sgn(\upsilon_{y'3})) + \frac{f_r cmg}{2}(sgn(\upsilon_{x'1}) - sgn(\upsilon_{x'2})).$$

Segundo Caracciolo, Luca, (1999), utilizando-se notação matricial, o modelo dinâmico reescrito no sistema de coordenadas inerciais é

$$\mathrm{M}_{q''} + C_{(q,q')} = E_{(q)}r,$$

sendo

$$\mathrm{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathrm{C}_{(q,q')} = \begin{bmatrix} cos\psi R_{x'} - sin\psi R_{y'} \\ sin\psi R_{x'} + cos\psi R_{y'} \\ Mr \end{bmatrix},$$

$$\mathrm{E}_{(q)} = \begin{bmatrix} \frac{cos\psi}{r} & \frac{cos\psi}{r} \\ \frac{sin\psi}{r} & \frac{sin\psi}{r} \\ \frac{c}{r} & \frac{c}{r} \end{bmatrix},$$

$$\tau = \begin{bmatrix} \tau_L \\ \tau_R \end{bmatrix} = \begin{bmatrix} \tau_1 + \tau_3 \\ \tau_2 + \tau_4 \end{bmatrix}, \tau_i = rF_{x'i}.$$

Incluindo a restrição não-holonômica no modelo dinâmico, como introduzido, tem-se:

$$\mathrm{M}_{q''} + C_{(q,q')} = E_{(q)}\tau + \mathrm{A}^T{}_{(q)}\lambda,$$

sendo λ o vetor de multiplicadores de Lagrange correspondente a equação. Para estimar as acelarações a partir do torque aplicado, devemos diferenciar a equação, substituindo λ, então teremos o seguinte resultado:

$$\acute{\eta} = (S^T MS)^{-1}S^T(E_\tau - MS'n\text{- C})$$

2.4.2 *LASER*

O sensor *LASER* utilizado também em robótica é um dispositivo responsável em medir valores de uma grandeza física ao redor de um robô, como por exemplo, a temperatura, a velocidade, a distância, a pressão e entre outras. Como um robô terá sua atuação em um ambiente real, podendo se deparar com obstáculos estático ou dinâmico é imprescindíveis sensores que lhes permitem ter informações em tempo real do ambiente que os cerca, independente do grau de luminosidade existente, fazendo com que essas informações possam ser interpretadas e processadas gerando respostas para com o ambiente.

O *LASER scanner* utiliza-se para avaliar o espaço livre em torno do robô através da medida da distância aos obstáculos presentes no ambiente circundante. O sensor emite um feixe de luz estruturada que viaja no espaço. Quando este feixe encontra um obstáculo é refletido de modo difuso e parte da luz regressa, sendo detectada pelo sensor que calcula o tempo de voo do feixe. Este tempo é proporcional, através da velocidade de propagação da luz no ar, à distância percorrida pelo feixe que é o dobro da distância do robô ao obstáculo que produziu a reflexão. Fazendo defletir o feixe horizontalmente através de um mecanismo de varrimento, é possível avaliar o espaço livre segundo um determinado ângulo.

A Figura 15 ilustra resultados experimentais da percepção do ambiente à frente de um robô usando um *LASER scanner* que tem uma resolução angular de 0.5º. Cada segmento de reta representa uma direção segundo a qual o feixe de luz é emitido a partir do *LASER scanner*. O comprimento do segmento representa a distância livre na direção correspondente.

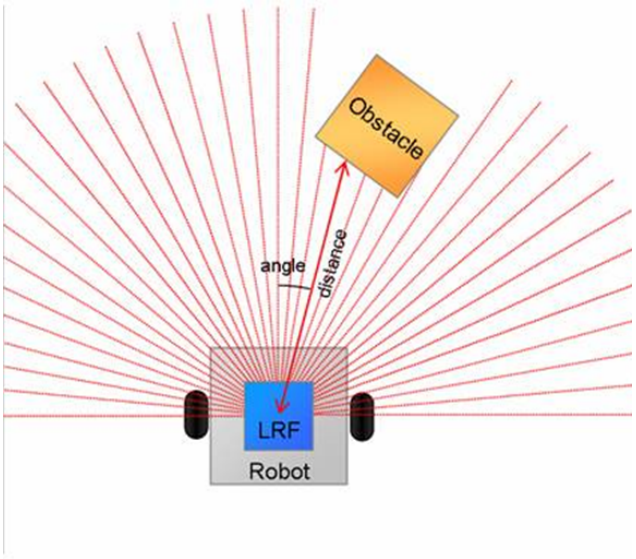

Figura 15 – Detalhamento dos feixes de *LASER* encontrando um obstáculo.

Fonte (FREITAS, 2008).

Os sensores *LASER*, que emitem uma fonte de radiação infravermelha e aguardam seu

retorno para a determinação da distância dos objetos, são classificadas como ativos.

Sensores de medição *LASER* são utilizados normalmente quando se deseja monitorar áreas, determinar medidas de distâncias, detectar e definir a posição de objetos. Portanto são sistemas de medição sem contato que mapeiam a região que os rodeiam em duas dimensões, outro ponto importante nesse sistema de medição é que todo o mapeamento não requerem refletores e nem marcações no espaço de trabalho.

2.4.3 Detecção de objetos móveis

A detecção de elementos móveis que podem ser objetos inanimados, animais, seres humanos ou até mesmo outros robôs é de fundamental importância para o sistema de manipulação em robótica, pode ser explicado de forma análoga com o sistema de visão humana e suas capacidades e características.

No sistema de visão humano, o processo de inicia-se como a captura da imagem através dos olhos e posteriormente diversa processos são realizados pelo Sistema Nervoso Central - SNC, que vão desde a identificação de retas e contornos (ZEKI, 1993) até a classificação de objetos e determinação da melhor forma de pegá-lo. Pesquisas apontam que os seres humanos apresentam postura de mãos especificas preferencias para pegar objetos de acordo como a forma predominante deste a tarefa a ser executada (CUTKOSKY, 1990).

Assim, além de determinar a posição de objetos e de classificá-los, através do sistema de visão os seres são capazes de identificar ou estimar outras propriedades importantes textura superficial, densidade, volume e momento de inércia. O sistema de visão também permite o mapeamento do ambiente, por exemplo, alguns segundos de observação de um ambiente são suficientes para que o mapa virtual seja construído pelo SNC, contendo informações com presença e classificação de objetos, pessoas, animais, plantas entre outros.

O sistema de visão também permite uma constante atualização do ambiente e detecção de modificações repentinas ou aproximação de objetos. Esses estímulos visuais são primeiramente detectados pelo sistema de visão periférico e são processados mais rapidamente que, por exemplo, a classificação de objetos ou mapeamento do ambiente. Quando um objeto se aproxima rapidamente de um observador as informações seguem um caminho diferenciado pelo SNC que permite uma resposta mais rápida, geralmente na tentativa de evitar ou amenizar os efeitos de uma colisão. Esta resposta reativa normalmente leva em torno de 200ms a 300ms e ocorre antes da classificação ou identificação do objeto (IBERALL, 1994).

Estas características observáveis do nosso SNC são importantes tanto para a percepção quanto para desenvolvimento de sistemas robóticos de manipulação ou rastreamento de objetos em movimento. Esses sistemas robóticos apresentam diversos sensores para esse tipo de percepção, como, por exemplo, os *LASERS* ou câmeras, a utilização de um deles ou até a fusão de dados dos dois, podem ser determinadas pelo grau de necessidade da aplicação ou pela resposta desejada.

Conforme foi mencionado anteriormente, a detecção de elementos na robótica móvel, que podem ser objetos inanimados, animais, seres humanos ou até mesmo outros robôs pode ser realizada por um tipo de sensor ou até mesmo a fusão de vários, sendo que o resultado dessa detecção é um uma imagem binária. Essa detecção de imagem pode ser feita por dois métodos de determinação de distância.

2.4.4 Métodos da diferença de fase

Com esta técnica é possível determinar a distância de um objeto através da defasagem (deslocamento de fase) entre dois feixes de *LASER*, um emitido e outro refletido. A Figura 16 apresenta um esquemático do arranjo utilizado para se medir a diferença de fase entre o feixe emitido e o feixe recebido enquanto a Figura 14 mostra duas ondas e a respectiva diferença de fase entre elas.

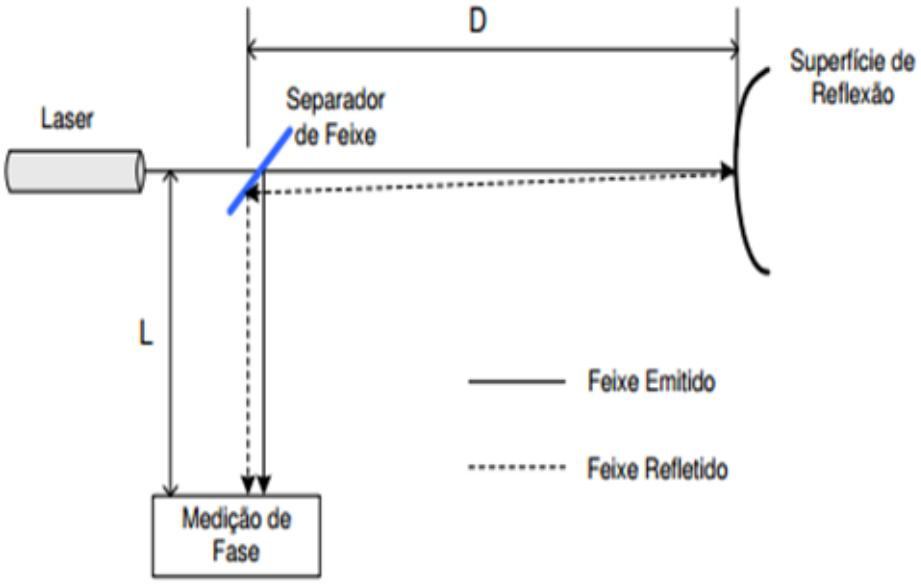

Figura 16 – Arranjo para medidas de distância através da diferença de fase.

Fonte: (FREIRE, 2002)

Para , D = 0 o feixe de referência (emitido) estará em fase com o feixe refletido. Para D > 0, a distância total percorrida pelo feixe de *LASER*, D', é dada por:

$$D' = L + 2.D\,(I)$$

ou

$$D' = L + \left(\frac{\alpha}{360}\right).\lambda\,(II)$$

Onde λ é o comprimento de onda do *LASER* e α é a diferença de fase entre o feixe emitido e o feixe recebido, como ilustra a figura 17.

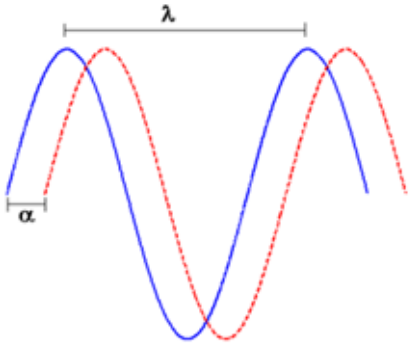

Figura 17 – Diferença de fase entre duas ondas.

Fonte: (FREIRE, 2002)

Mas, se $\alpha = 360\circ$, os dois sinais estão alinhados e não é possível diferenciar D'= L de D' = L + nλ, para n=1, 2, 3 . . .

Assim, uma solução única somente pode ser obtida se $\alpha < 360\circ$, o que implica que D< 2λ.

Igualando as Equações (I) e (II), a distância entre o sensor *LASER* e um objeto é:

$$D = \left(\frac{\alpha}{360}\right) . \left(\frac{\lambda}{2}\right)$$

Entretanto, este método é impraticável para medir distâncias maiores que a metade do comprimento de onda do *LASER*. Para um *LASER* de hélio-neon, cujo comprimento de onda é 632,8nm, a maior distância que pode ser medida é 316,4nm. Uma solução simples para este problema é modular a amplitude do *LASER* com uma onda senoidal de alta frequência.

Por exemplo, para uma onda moduladora de 10 MHz, o comprimento de onda é de 30m, o que possibilita uma medição de distância de até 15m. Uma onda senoidal modulada em amplitude pode ser vista na Figura 18.

Assim, o feixe modulado é emitido e, ao retornar, ele é demodulado e comparado com feixe de referência para determinar o deslocamento de fase.

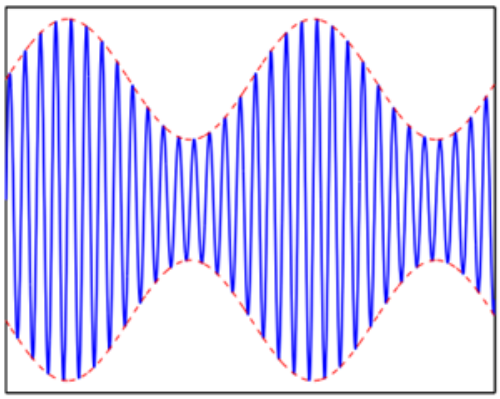

Figura 18 – Onda modulada em amplitude.

Fonte: (FREIRE, 2002)

2.4.5 Método do tempo de voo

Neste método, um pulso de radiação é emitido pelo transmissor. Este pulso, ao encontrar uma superfície, retorna até o receptor. A medida de distância entre o sensor e a superfície refletora é determinada pela multiplicação da velocidade do pulso (no caso do *LASER* esta velocidade é a velocidade da luz), pelo tempo que este levou para sair do transmissor e chegar até o receptor, ver Figura 19.

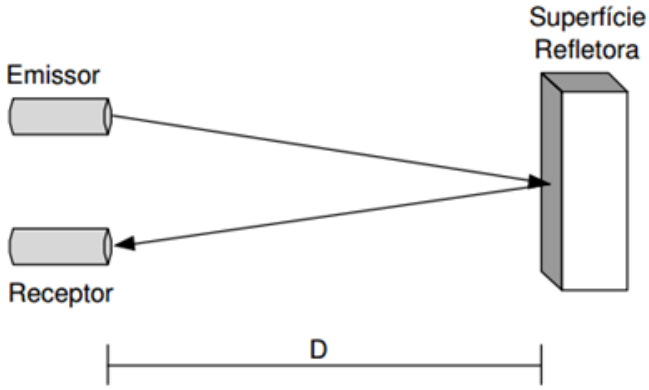

Figura 19 – Princípio da técnica do tempo de voo.

Fonte: (FREIRE, 2002)

Assim sendo, a distância entre o sensor e o objeto, D, é dada por:

$$D = c.\left(\frac{\Delta t}{2}\right)$$

Onde Δt é o tempo de propagação do pulso entre o transmissor e o receptor, c é a velocidade da luz no meio em que se propaga e D é a distância entre o instrumento de medição e a superfície refletora.

Considerando que a velocidade da luz no ar, c, é igual a 3 × 108 m/s, o tempo entre o pulso emitido e o recebido é muito pequeno. Portanto, para se efetuar medidas de distância a partir deste método, os equipamentos utilizados devem ser altamente precisos e consequentemente com um custo elevado.

2.4.6 Método de subtração de imagem

Existem inúmeras proposta de métodos e algoritmos para essa técnica de subtração de fundo de cena, conhecida também por *blackground subtraction*[11]. Essa técnica consiste em detectar um conjunto de feixes que são enviados por um *LASER* ou um vídeo e posteriormente um primeiro plano de uma imagem é extraída para processamento posterior (reconhecimento de objetos etc.) Geralmente as regiões de uma imagem de interesse são objetos (humanos, carros, texto etc) em seu plano. Após a etapa de pré-processamento de imagem (que pode incluir a filtragem, pós-processamento, como morfologia, etc), após a localização é exigido o uso desta técnica. A subtração é uma abordagem amplamente utilizada para a detecção de objetos em movimento. A lógica na abordagem é o de detectar os objetos em movimento a partir da diferença entre o quadro atual e um quadro de referência, muitas vezes chamado de "imagem de fundo", ou "modelo de fundo". A subtração é feito principalmente se o objeto em questão é uma parte de um do ambiente.

2.4.7 Arquitetura de rôbos móveis

A arquitetura quando relacionada ao software do robô móvel, descreve a maneira com o controle inteligente é construído, apresentando quais os módulos presentes e as formas de integrações entre eles (GRASSI, 2006). No entanto um robô móvel não é somente um conjunto de *hardware* e *software* combinados para um determinado fim, o robô é um sistema de controle que organiza todos os seus recursos e limitações como finalidade de executar uma tarefa pré-determinada. Dependendo da aplicação os sistemas de controle robótico podem ser simples ou muitos complexos. Várias são as formas que determinam esse grau de complexidade, sendo um dos mais importantes, a maneira como o robô vai se relacionar com o ambiente que ele está inserido, pelo fato que ele sofrerá influência e também influenciará. Suas entradas e saídas são muitos variáveis e, organizá-la, aumenta a complexidade do desenvolvimento do sistema (MAETA, 2011).

Entre as diversas características de arquiteturas de *software* para robôs móveis, o que se diferencia é a forma de raciocínio. Na robótica móvel raciocínio está ligada a forma como o sistema reage com o meio ambiente, seja por meio da leitura de seus sensores ou a entrada de dados especificando um objetivo. Existem três grandes arquiteturas para a robótica móvel que podem ser dividida em Reativa, Deliberativa e a Hibrida.

[11] Subtração de fundo

2.4.8 Arquiterutas reativas

A arquitetura Reativa é formada por uma série de comportamentos que fazem relações sensoriais a um conjunto de ações do robô, além de ser composta por métodos que possibilitam uma correta integração destes comportamentos(PIERI, 2002).

A principal característica da arquitetura Reativa está relacionada com a possibilidade de programar um sistema de controle que tenha uma resposta imediata a uma reação de modificação do meio ambiente, fazendo com que o robô possa operar em ambientes dinâmicos. Essa velocidade de reação está relacionada com a simplicidade no tratamento do sinal dos sensores e é uma forma direta pela qual a percepção, ou estimulo, está associada com uma ação ou uma reposta (GRASSI, 2006)..

O arranjo em uma arquitetura reativa pode ser feita em duas formas a competitiva ou cooperativa. Na coordenação competitiva os comportamentos ativos que o robô vai assumir será apenas um, obedecendo sempre uma hierarquia. Na coordenação cooperativa todos os comportamentos ativos contribuem para a ação do robô.

A figura 20 resume essa arquitetura também denominada pelo conceito de ação-reação, no qual sua ação é relativa aos estímulos recebidos do ambiente. Os estímulos gerados e percebidos pelo robô não serão processados para uma avaliação e elaboração de um plano, nessa arquitetura não a fase de planejamento, a saída é totalmente relacionada entrada.

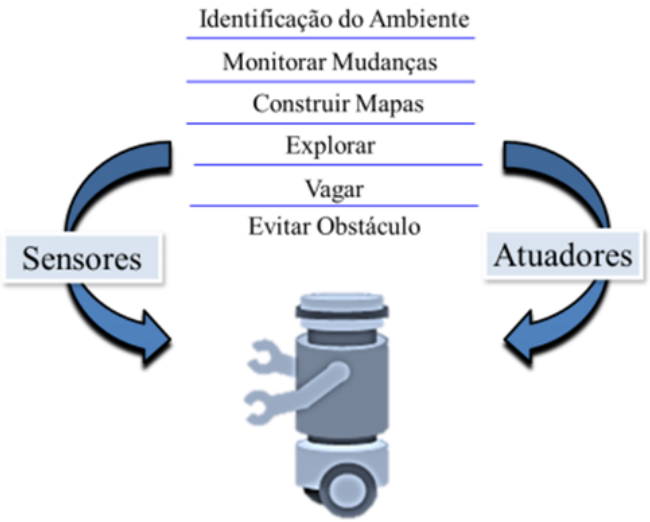

Figura 20 – Representação da arquitetura reativa.

Fonte: (WOOLDRIDGE, 1996)

2.4.9 Arquiteturas deliberativas

Essa característica corresponde a todo o planejamento e a tomado de decisão que robô faz para que o venha executar uma determinada atividade (GRASSI, 2006). Essa arquitetura deliberativa realiza suas ações internamente de maneira lógica e sequencial, primeiro percebendo o ambiente, por meio dos sensores ou dispositivos capazes de perceber

o mundo ao redor, depois planejam a ação a ser tomada e, por fim, ocorre a ação a ser tomada.

Conforme a figura 21, uma tarefa é subdividida e a execução se dará uma por vez, a reorganização e junção dessas subdivisões delas vai caracterizar o objetivo do sistema robótico. Esta arquitetura apresenta pontos positivos em ambientes muito dinâmicos e que podem ocorrer situações inesperadas, forçando o sistema a perceber e planejar novamente uma nova ação ser tomada.

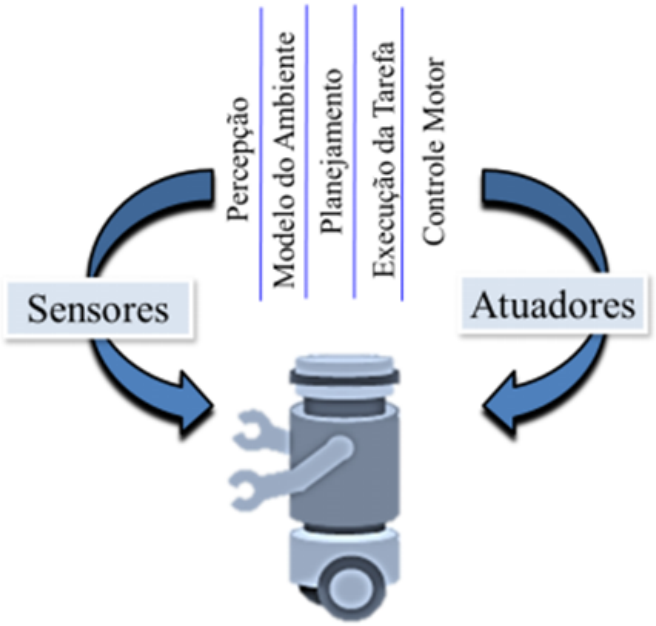

Figura 21 – Representação da arquitetura Deliberativa.

Fonte: (WOOLDRIDGE, 1996)

Esse modelo pode ser do tipo simbólico ou geométrico. Sendo que o simbólico é baseado utilizando lógica, muitas das vezes a inteligência artificial é empregada. O modelo geométrico utiliza-se como princípio representações de espaços livre e regiões de obstáculos. Porém o modelo geométrico é o mais utilizado para projetos que robótica móvel devido às atividades que os robôs autônomos executam em ambientes não-planejados (GRASSI, 2006).

Desta forma o raciocínio deliberativo torna-se limitado em ambientes estáticos e controlado devido à dependência com o ambiente que o robô esta inserido. Ainda tem algumas características que devem se levar e, conta utilizando esse tipo de raciocínio que é a mudança de ambiente, pois é necessário que seja percebido, para que não haja possíveis riscos na execução das atividades.

2.4.10 Arquiteturas híbridas

A arquitetura híbrida é a junção das duas arquiteturas citadas anteriormente, partindo do conceito que certas atividades exercidas por um robô podem precisarem de um planejamento e em outras simplesmente a responder a estímulos do ambiente. Na figura 22 é possível observar que essa estrutura híbrida o sistema continua recebendo informações dos sensores e enviando para os atuadores, porém essas informações podem estar ligadas tanto a módulos deliberativos como a módulos reativos.

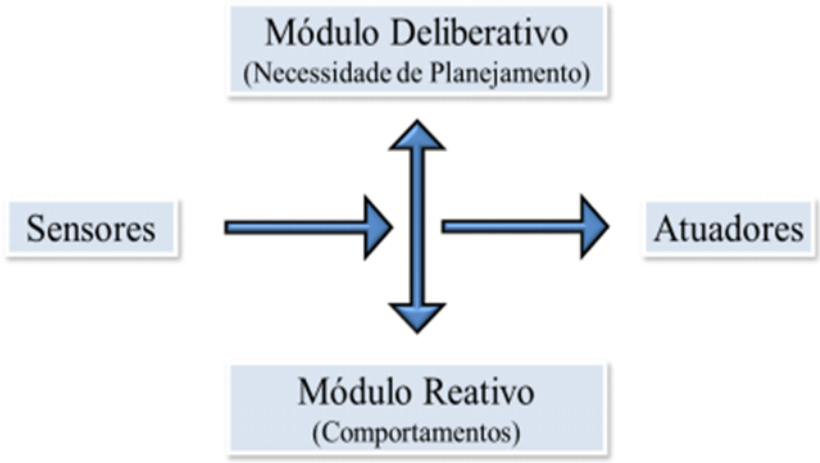

Figura 22 – Representação da arquitetura híbrida

Fonte: (WOOLDRIDGE, 1996)

2.5 Trabalhos relacionados

Na *Wuhan University of Technology*[12] foi desenvolvido um projeto, por Bin Lei e Wenfeng Li (LEI; LI, 2007), no ano de 2007 que fazia uma abordagem ao tema de desvio de obstáculos com a utilização de um *software* que empregava a lógica *fuzzy* para o desvio de obstáculos. Esse controle de desvio de obstáculos é obtido com as entradas vinda dos sensores indicando um possível obstáculo critico e sua localização aproximada. A localização do obstáculo fornece a informação que o sistema *fuzzy* necessita para que por meio de sua base de conhecimento possa produzir uma saída que é como reagir ao obstáculo em questão. Esse tipo de abordagem utilizada para navegação autônoma é bem útil quando se trata para desvios de obstáculos por existir um numero grandes de incertezas presentes no ambiente de navegação do robô, porém faltaram estudos para adicionar novos comportamentos.

Outro trabalho que fazia abordagem utilizando o desvio obstáculos com a utilização de uma câmera foi desenvolvido no *Automation Science and Engineering*[13] (KUO; SYU,

[12] *Universidade de tecnologia de Wuhan*
[13] *Ciências da engenharia e automação*

2011) na Itália no ano de 2011. Esse trabalho consistia em uma cadeira com rodas que tem autonomia de navegação e que desvia de possíveis obstáculos a sua frente em tempo real, porém a utilização de um sensor câmera deixa o projeto com limitação, isso por que qualquer variação na taxa de luminosidade causaria interferências na interpretação dos dados, que posteriormente seriam processados para que seja tomada a ação de desvio e movimentação do robô.

O trabalho desenvolvido, na universidade de Alcalá na Espanha, por Guillermo Asín Prieto e Julio Pastor Mendoza(GUILLERME E MENDOZA, 2013) no ano de 2013 trata-se do desenvolvimento de uma plataforma didática baseado no *Player/Stage*, possibilitando desta forma que aluno consigam desenvolver estudos relacionados ao campo da robótica móvel com um custo baixo, possibilitando também que alunos desenvolva algoritmos e possam testar. Porém todo o desenvolvimento é voltado a situações de robôs seguidores de linha descartando o uso do *LASER* para fazer a percepção do ambiente no qual ele está inserido.

Vários foram os trabalhos envolvendo Ultra-Som como sensor de detecção de obstáculos para navegação de rôbos móveis, com por exemplo o trabalho apresentando no jornal de robótica em 1987 que tratava de um mapeamento e navegação mundo-real (ELFES, 1987) . Outro trabalho foi apresentado em 2004 na Universidade de Espírito Santo que tratava do desvio tangencial de obstáculos em ambientes não-estruturados (FERREIRA, 2004). Porém, algumas situações em que os sensores ultra-sônicos encontram dificuldades em detectar certos tipos de obstáculos, como é o caso de uma quina ou de uma superfície com inclinação muito grande em relação ao eixo de emissão do sensor ultra-sônico.

Em 2005 foi apresentado um trabalho no simpósio de automação inteligente na universidade de São Luiz do Maranhão que utilizava o fluxo óptico calculado numa imagem omnidirecional para detectar os obstáculos que existem no meio (FRANCA, VASSALLO, 2005). Também utilzando o fluxo óptico outro trabalho que trata da divergência estério para navegação de robô móvel em Nova Iorque no no ano de 1993. Porém, esses tipos de sensoriamento que se baseia na visão computacional é bastante sensível aos efeitos de iluminação. Além disso, o uso da visão requer um elevado custo computacional. Isto pode favorecer o uso da visão computacional para sistemas com maior capacidade de processamento, porém não impede que sistemas com menor capacidade de processamento usem este tipo de sistema sensorial.

3 Materias e métodos

3.1 Estudos sobre a robótica móvel

Para um melhor entendimento da simulação abordada nesse trabalho, foi necessário um estudo sobre os conceitos básicos de robótica móvel. A introdução teórica do capitulo 2 é o fruto desse estudo.

3.2 Estudos de métodos para o simulador *Player/Stage*

Existem diversos simuladores de robôs móveis atualmente disponíveis no mercado. Alguns desses simuladores exigem uma licença para sua utilização, podendo ser inclusive comercializados juntamente com alguns modelos de robôs comerciais, enquanto outros podem ser utilizados livremente. Outra característica importante presente em alguns simuladores é a integração com programas que permitem controlar diretamente os robôs reais. Dessa forma, os programas de controle testados no simulador podem ser facilmente portados para uso com robôs reais.

Foi verificado utilizando referências bibliográficas que o simulador *Player/Stage* e o *ROS/Gazeebo* têm sua utilização bem difundida em meios acadêmicos, porém este último possui formulações relativamente mais complexas e por questão de simplicidade de implementação foi adotado o ambiente de desenvolvimento *Player/Stage.*

3.3 *Player*

O *Player* é um sistema de código aberto e livre distribuição que teve início de seu desenvolvimento no ano de 2000 por pesquisadores da universidade do sul da Califórnia, seu principal objetivo era suprir a falta de um sistema que fosse capaz de simular e controlar robôs móveis, pois até então não havia um sistema eficiente e com um grande poder de compatibilidade e que fugisse dos modelos arquitetônicos de programação em robôs. Este sistema foi amplamente difundido pelas universidades do mundo e vem sendo desenvolvido pesquisas para o aprimoramento que adequem a um número maior de plataformas robóticas e sensores comerciais (COLLETT,2005).

O *Player* tem como aplicação distribuída o modelo cliente/servidor, que faz a distribuição das tarefas e cargas de trabalhos. O servidor tem como função de interligar a interface do robô com os sensores, obtendo valores, enviando e recebendo instruções do cliente para realizar o controle do robô e dos sensores. O cliente é o programa que controla

o robô e é responsável por obter os dados do servidor, fazer a interpretação e enviar as instruções para que o robô possa executar a tarefa desejada.

Este modelo de arquitetura baseado no cliente/servidor torna-se viável no controle de robôs móveis, pois um cliente é capaz de controlar diversos servidores e outros diferentes clientes quem podem controlar diversos sensores de um mesmo robô. A comunicação é feita por protocolos de controle de transmissão e protocolos de interconexões. O cliente *Player* foi projetado para ser compatível com várias linguagens de programação, com o suporte principal aos ambientes C, C++, *Python*.

3.4 *Stage*

O *Stage* é um simulador, de livre distribuição, de robôs e sensores para trabalhar em ambientes de duas dimensões que seja compatível com o *Player*, podendo desta forma fazer a simulação de diversos robôs móveis e sensores, controlado por um ou mais clientes, a figura 23 ilustra um ambiente típico de trabalho. Sua autenticidade em fazer simulações em ambientes reais faz dele bastante confiável ao ponto de ser testado em robôs reais, modificando somente o protocolo de interconexões. Sua comunicação também é feita por protocolos de controle de transmissão e protocolos de interconexões.

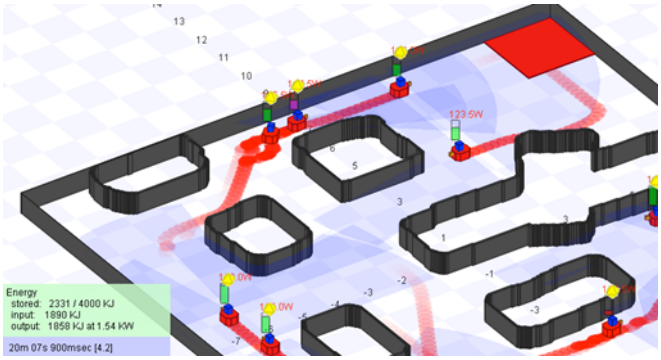

Figura 23 – Ambiente do simulador Stage

Fonte: https://sites.google.com/site/mecaelecktro/home/mapeamento-2d. (Acesso: 20/08/2014)

3.5 Outros simuladores

3.5.1 *Webots*

Webots desenvolvido pela *Cyberbotics* Ltd e o *Microsoft Robotics Developer Studio* que como seu nome já diz é desenvolvido pela Microsoft.

Os motivos para a escolha dos softwares do projeto Player em detrimento dos outros que serão apresentados abaixo são:

- Os softwares do Player possuem código aberto, o que possibilita adaptações no código caso necessárias;
- As ferramentas são gratuitas não sendo necessário nenhum tipo de licença;
- Os softwares rodam em um sistema operacional gratuito de larga escala que é o Linux;
- Diversos projetos já foram desenvolvidos no mundo fazendo o uso destas ferramentas com sucesso;
- Grande comunidade desenvolvedora oferecendo suporte.

O *Webots*, assim como o *Player/Stage*, é um simulador robótico profissional usado largamente para propósitos comerciais e educacionais. O projeto *Webots* teve início em 1996 no *Swiss Federal Institute of Technology*[1] em *Lausanne* na Suíça. Anos depois, com o sucesso do projeto, este *software* passou a ser desenvolvido comercialmente e hoje é controlado pela *Cyberbotics* Ltd.

As semelhanças com o *Player/Stage* são muitas. De acordo com commercial mobile robot simulation software, o Webots inclui um conjunto de sensores e atuadores frequentemente usados em experimentos robóticos e também um conjunto de modelos de robôs que podem ser livremente modificados. É possível ainda utilizar o *Webots* para construir novos modelos aonde se define suas propriedades gráficas que incluem o formato, dimensões, cores e texturas e suas propriedades físicas que consistem em massa, fatores de fricção e até mesmo constantes de mola e amortecimento. Além disso, o *Webots* faz uso também da *Open Dynamics Engine*[2] (ODE) para simular a dinâmica de corpos rígidos.

Os programas para controle de robôs que se comunicam com o *Webots* podem ser escritos em uma variedade de linguagens de programação como C, C++, Java e *MATLAB* e o *Webots* funciona nos sistemas operacionais mais comuns da atualidade incluindo aí o *Windows, Mac OS X* e *Linux.*

A maior desvantagem do *Webots* é o fato de este ser um *software* pago de alto custo. Ele é vendido hoje em dia em duas versões, sendo uma educacional dedicada ao ensino de robótica em salas de aula e uma versão profissional mais completa dedicada ao desenvolvimento com intuito comercial.

3.5.2 *Microsoft Robotics Developer Studio - MRDS*

O *Microsoft Robotics Developer Studio*, muito conhecido por sua sigla MRDS, é um ambiente para controle e simulação de robôs totalmente baseado na plataforma *Windows.* De acordo com desenvolvedor do simulador, o MRDS faz uso da *Concurrent and*

[1] *Instituto federal de tecnologia da Suiça*
[2] *Engenharia de dinâmica aberta*

Coordination Runtime[3] (CCR), uma biblioteca de programação concorrente baseada na plataforma *.NET* da Microsoft. Esta biblioteca faz o controle de tarefas paralelas usando troca de mensagens o que possibilita a coordenação de múltiplos serviços para atingir comportamentos complexos como são os dos robôs.

O MRDS inclui uma ferramenta de programação visual usada para criar e depurar aplicações para robôs, interfaces baseadas em Windows e web, simulação em três dimensões e acesso simples a sensores e atuadores. Estas são características em geral comuns a todos as ferramentas de desenvolvimento de robôs existentes. Porém, diferentemente do *Webots* e do *Player/Stage*, o MRDS faz uso da biblioteca chamada *PhysX* para simulação da dinâmica de corpos rígidos.

Porém, ainda há uma grande limitação em relação ao *Player/Stage*: o fato de não possuir código aberto impossibilita a modificação de código, que é muitas vezes necessário para adaptações de projeto.

3.6 Robôs *Pioneer P3-AT*

O robô *Pionner P3-AT* é uma plataforma robótica desenvolvida na empresa *Mobile Robots*, sua unidade possui quatro rodas, um computador embarcado e um *software* compatível com todos os robôs da *Mobile Robots*. Como ilustra a figura 24. Por possuir uma estrutura confiável, versátil e possuir uma durabilidade resistente em seu desempenho o *P3-AT* se adapta em vários tipos de terrenos, torna-se popular em projetos ao ar livre.

O *P3-AT* suporta a conexão de diversos tipos de dispositivos externos como câmera, braços mecânicos, *LASER,* serial *Athernet* para operação remota, altos falantes e microfones e muitos outros. Quando sua estrutura está adaptada com o pacote de navegação a *LASER* o robô pode mapear armazéns, galpões, espaços de laboratórios e qualquer espaço interior e exterior em minutos por possuir uma mobilidade de navegação de aproximadamente 1.6 metros por segundo.

O *Pionner P3-AT* é uma base externa para muitos fins, usada para aplicação e prototipagem que envolve mapeamento, navegação, monitoramento, reconhecimento de visão e outros comportamentos.

[3] *Simulação e coordenação de tempo de execuçao*

Figura 24 – Robô *Pioneer P3-AT*

Fonte: (Manual)

Características e benefícios do *Pionner P3-AT:*

- Confiável - Construção é durável e resistente. Adapta-se facilmente a pequenos espaços, resistente a colisão e ou outros obstáculos que outras plataformas robóticas têm dificuldades em trabalhar;
- Personalizável – São encontrados vários acessórios compatíveis e testados que integrar-se com a sua plataforma robótica, assim com o suporte adicional está disponível para atualizações;
- Plataforma de Referência - Por ser um padrão de plataforma móvel inteligente de robôs é possível encontrar diversas revistas, artigos que faz referencia a essa arquitetura, assim como exemplos de plataformas aplicadas em pesquisa.

3.7 *LASER SICK* LMS200

Este sensor fornece a medida de distância através da medição do tempo de voo dos pulsos dos feixes de *LASER*, os qual são emitidos e, se um objeto for encontrado, refletidos. A figura 25 ilustra esse *LASER SICK LMS200.*

Figura 25 – *LASER SICK LMS200.*

Fonte: (Manual)

O feixe de *LASER* é deflectado por um espelho rotativo interno, realizando uma varredura de 180 ◦ , em forma de leque, conforme a figura 26 da área os arredores do sensor, As medidas de distância podem ser tomadas em intervalos de 1 ◦ , 0.5 ◦ e 0.25 ◦ . Esta escolha é feita pelo software, o contorno de um alvo pode ser obtido por meio de uma sequência de impulsos recebidos cujas medidas são disponibilizadas via uma interface serial.

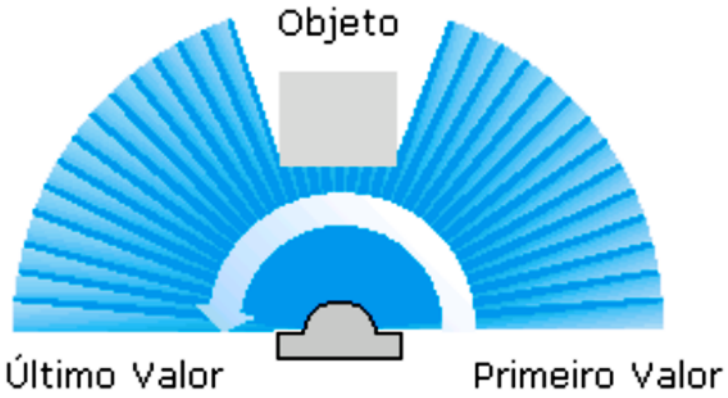

Figura 26 – Varredura do *LASER LMS200* em 180 ◦

Fonte: (Manual)

A Figura 27 ilustra o feixe de *LASER* emitido e o respectivo feixe refletido por um objeto. A distância que o sensor se encontra deste objeto é determinada a partir destes dois feixes de *LASER.*

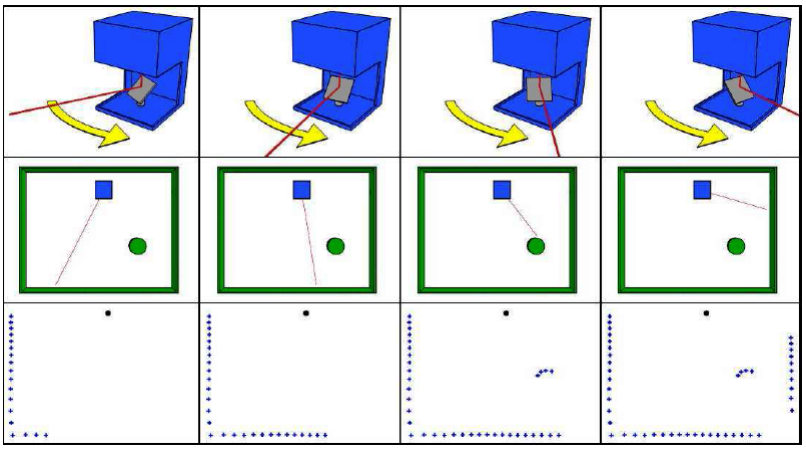

Figura 27 – Princípio de operação do *LASER LMS200*.

Fonte: (Lidar 2009)

Outras características do *LASER SICK LMS200* podem ser vista na tabela 1.

Tabela 1 – características do *LASER SICK LMS200*

Alcance	150 m
Resolução	10 mm
Resolução angular	0.25 ◦ , 0.5 ◦ , 1.0 ◦ via software
Tempo de resposta	52, 26, 13 ms dependendo da resolução
Tensão de alimentação	24 V
Temperatuda de operação	0 ◦ C a 50 ◦ C
Dimensões e pesos	155 x 185 x 156 m (L, A, P)
Peso	4.5 Kg

Foram vistos neste capítulo as ferramentas necessárias para a implementação do projeto. o *Player/Stage* tem um papel muito importante devido o poder que ele tem de simular o robô *Pionner* juntamente com o *LASER SICK LMS200* ambos integrados com os mesmo objetivos de rastreamento.

4 Experimento

Neste capítulo será explanada toda a execução da simulação virtual de navegação autônoma, mostrando cada etapa da máquina de estado até que o algoritmo venha atingir seu objetivo final. É importante destacar que este trabalho deve levar em consideração uma série de requisitos e componentes, com, por exemplo:

- Tipo de tarefa do robô: Rastreamento de objetos em movimento;
- Tipo e precisão dos sensores embarcados: *LASER SICK LMS200* com uma capacidade de detecção de obstáculos com capacidade de detectar obstáculos em um *range* máximo de $25m$ a $150m$ de distância (fecho *LASER* estreito e direcional) com precisão de até 10mm (erro estatístico de 5mm), ângulo de abertura de 100 a 180 graus e tempo de resposta de entre 10 e 50ms;
- Tipo e precisão dos atuadores: Acionamento diferencial (2 motores independentes), robô de características holonômicas (sem restrições de graus de liberdade, podendo girar e se deslocar em qualquer direção), e podendo atingir velocidade de até 1.6m/s;
- Desviar de obstáculos: detectar obstáculos e poder assim evitar a colisão com os mesmo, preservando a integridade do robô e dos elementos externos;
- Mapeamento do ambiente: exploração e construção de um mapa do ambiente, onde o robô é capaz de realizar uma reconstrução digital do mapa do ambiente, identificando onde existem os obstáculos fixos e móveis no ambiente gerando, por exemplo, um mapa de ocupação do ambiente;
- Planejamento ações: uma vez definida a tarefa a ser realizada pelo robô móvel, é possível estabelecer um plano de ações, que pode ser composto da execução de sub-tarefas mais elementares se deslocando em uma direção-alvo em movimento desviando de obstáculos.
- Auto-Localização: determinar a localização do robô no ambiente (posição e orientação), com ou sem o uso de um mapa do ambiente, de modo a poder planejar e executar o deslocamento seguindo uma determinada trajetória;

Para criar um modelo para simulação virtual é necessário elaborar um modelo virtual simplificado da realidade, implementando este em um ambiente computacional. O modelo simplificado deve desconsiderar o que não for necessário e/ou relevante para efeitos da simulação dos componentes reais do robô. Entretando, este modelo deve intergrar os elementos e caracteristicas que são mais relevantes para o funcionamento do sistema robótico, o que usualmente na área de robótica significa modelar:

- Sensor: o sensor selecionado para o desenvolvimento deve ser passível de simulação, onde, por exemplo, no caso dos sensores de medida de distância, o modelo criado deve levar em consideração o alcance (menor/maior distância medida), a precisão, e questões relativas ao direcionamento no processo de detecção de obstáculos. Também deve ser modelado o erro/ruído típico do sensor em questão, sendo que sabidamente não existem "sensores perfeitos";

- Atuador: deve ser modelado, de forma a simular as ações realizadas pelo robô. O atuador a ser modelado é responsável pela locomoção do robô móvel, no qual também é importante que o modelo implementado considere a precisão/erros associados às ações. Por exemplo, ao se comandar um giro do robô de N graus, muito provavelmente haverá um erro associado a este comando, resultando em uma nova orientação próxima a desejada, na qual este não deverá atingir exatamente o ângulo de orientação previsto;

- Comportamento Físico do Robô: usualmente não é necessária uma modelagem individual de cada atuador do robô, podendo ser integrado um modelo de comportamento físico deste, através de um modelo cinemático do mesmo. Deste modo será possível prever a trajetória que será executada pelo dispositivo móvel a partir do modelo físico/comportamental.

Tendo como base de estruturação de algoritmo uma máquina de estado, que nada mais é que uma modelo matemático usado para representar programas. A máquina de estado está presente agindo somente um estado por vez, este estado fica caracterizado como estado atual, onde cada estado armazena informações de acontecimentos passados e os transmitem em ações de mudança deste para o outro. Essa transmissão é a indicação que assinala a passagem de um estado atual para outro.

Para o desenvolvimento da máquina de estado a ser implementado no algoritmo de rastreamento, foram levados em consideração três estados. O fluxograma conforme ilustra a figura 28 faz uma representação similar ao do projeto, levando em consideração os três modos de estados que são o Modo Escanear, Rastrear e Vagar.

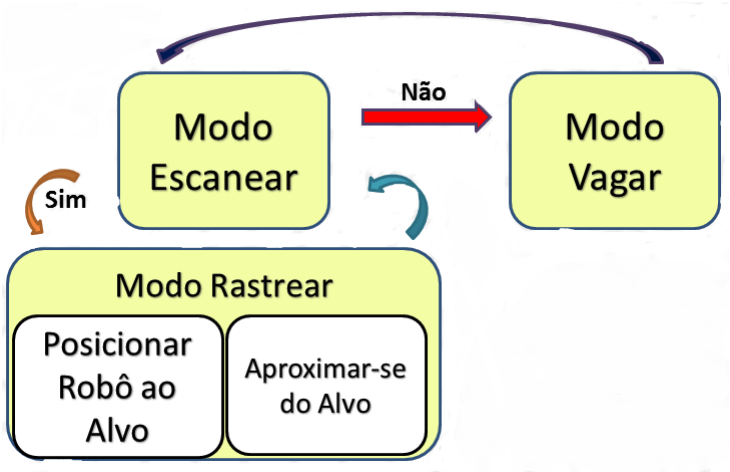

Figura 28 – fluxograma da máquina de estado

Fonte: (Próprio autor)

Observa-se que existem duas funções que fazem parte do Modo Escaner, uma para ajuste de ângulo do robô com o do alvo e outra para aproximar-se. Essas funções que tem como base o posicionamento do robô com o mapa, denominando-se assim de posicionamento global (relação da posição do robô com o mapa) e posicionamento local (relação da posição do robô com o Laser).

4.1 Desenvolvimento do algoritmo

O algoritmo para rastreamento de objetos dinâmicos foi implementado e validado conforme o algoritmo descrito no apêndice A, dispondo-se dos materiais e métodos propostos no capítulo anterior. A fim de proporcionar uma melhor compreensão da resolução proposta para o problema, um pseudocódigo é descrito no Algoritmo 1:

Algoritmo 4.1 Algoritmo para Rastreamento

```
1: ConfigParam();
2: IniciaParam();
3: while true do
4:     if estado = ESCANEANDO then
5:       vel_lin ←0;
6:       [Rastrear, θ_d, dist] = SubtrBckGnd(Amostras);
7:       if Rastrear = true then
8:          estado = RASTREANDO;
9:       else
10:          estado = VAGANDO;
11:      end if
12:    else
13:      if estado = RASTREANDO then
14:         CorrigeRefs(θ);
15:         velrot = Alinha(θ,θ_d, Δt);
16:         vellin = Aproxima(dist, tempo);
17:         estado = ESCANEANDO;
18:      else
19:         if estado = VAGANDO then
20:            [vel_lin,vel_rot] = DesviaObstaculos(T);
21:            estado = ESCANEANDO;
22:         end if
23:      end if
24:    end if
25:    InputPioneervel_lin,vel_rot);
26:    Atrasa(tempo);
27: end while
```

Basicamente, uma máquina de estados é construída no algoritmo 1. Este, por sua vez, é constituído de um laço principal onde é feita a comutação dos estados que, por fim, alimentam as entradas de referência de velocidade de rotação e linear.

Nas preliminares do programa são feitas as configurações e inicializações de constantes e parâmetros, executadas através das funções *ConfigParam* e *IniciaParam*, respectivamente. Estas funções têm por objetivo carregar, com valores pré-estabelecidos ou calculados, as constantes utilizadas durante todo o algoritmo. Além disso, alocam espaço de memória e declaram as variáveis necessárias para a compilação e execução do algoritmo. Dessa maneira, garante-se velocidade nula e chaveamento para o primeiro estado durante esta fase de preparação.

Uma vez que o programa entra no laço de comandos principal, todo o controle do robô

será definido através dos estados RASTREANDO, VAGANDO e ESCANEANDO. Este último é o estado definido nos preâmbulos como iniciador a máquina de estados.

O estado ESCANEANDO tem por objetivo realizar uma verificação do ambiente a procura de objetos dinâmicos. Essa tarefa é executada através da técnica de subtração de *background*, onde o algoritmo faz a interpretação dos feixes refletidos do *LASER SICK LMS200* obtendo-se informações do ambiente e fazendo a comparação de uma primeira amostra como a suas posteriores. Desta forma a subtração do fundo consiste na diferença espacial entre o modelo do ambiente, a imagem atual captada e a diferença temporal entre imagens, o que consiste na subtração dos feixes refletidos de quadros sucessivos. A quantidade de verificações necessárias, representada no algoritmo por amostras é computada nos preâmbulos. Resultados empíricos durante a etapa de validação do algoritmo indicaram que um valor entre três e cinco supriria as necessidades de projeto. A detecção adequada de um obstáculo móvel conduz a máquina de estados para seu segundo estado denominado RASTREANDO. Caso contrário, o estado VAGAR representa o próximo destino do programa. A necessidade de um filtro para pós-tratamento do sinal foi provada ser necessária para a função *SubtrBckGnd* e este problema será discutida em maiores detalhes na seção 5.1.

O estado RASTREANDO é chamado imediatamente após uma verificação de um objeto móvel no ambiente e, para realizar a tarefa de rastreamento, o robô deverá se aproximar do alvo detectado a partir de métricas definidas. Para o robô *Pioneer*, as referências de ângulo para o ambiente, figura 29, e para o robô são completamente distintas conforme visto na revisão bibliográfica e, portanto, foi necessário aplicar uma função de correção de referências globais e locais para então tratar o problema de alinhamento. Esta correção foi feita através da função *CorrigeRefs*.

Ainda no estado RASTREANDO é executada a tarefa de alinhamento com o objeto desejado através da função alinha, figura 29. Este alinhamento consiste em girar o robô em torno do próprio eixo a fim de ficar frente a frente com seu alvo para então realizar a tarefa de aproximação. A aproximação é feita tendo como base a distância ao alvo e o quão perto pretende-se chegar através da função Aproxima. As tarefas de alinhamento e aproximação são feitas com base no tempo de atraso e nos parâmetros conhecidos e computados pelo sensor.

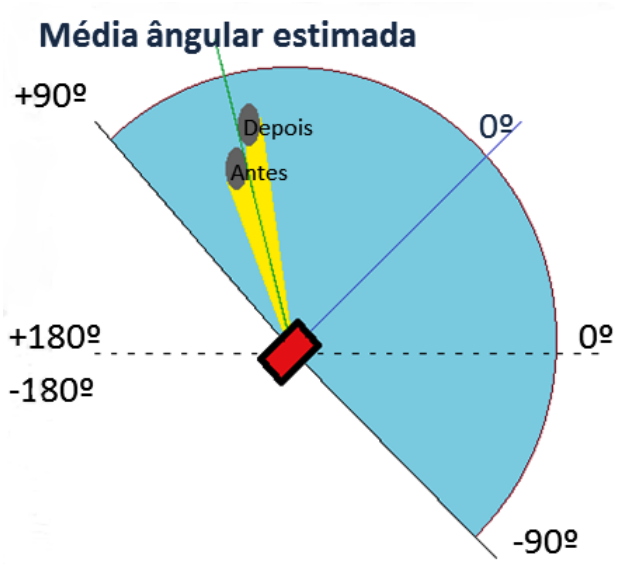

Figura 29 – Implementação - Alinhamento

Fonte: (Próprio autor)

Para essa implementação do alinhamento do robô com o alvo é utilizado a função *CorrigeRefs* seguindo os seguintes passos:

1. O algoritmo faz uma estimativa da média angular compreendida entre a área do primeiro ponto onde foi detectado o objeto com o último ponto onde o mesmo foi visto. Isso considerando somente sua referência local;

2. Fazer a correção ângular do robô com a referência global, desta forma o algoritmo identifica que caso o ângulo da sua normal não esteja alinhado com a referência global, faz-se necessário, acrescentar esse valor de ajuste;

3. Com esses valores de referências, então é feita a soma que corresponderá a um valor de ângulo total que o robô terá que executar fazendo o giro em seu próprio eixo.

Após a etapa de alinhamento o robô faz a aproximação do alvo. Por se tratar de um objeto em movimento o robô irá fazer seu deslocamento em linha reta para o ponto estimado de onde foi localizado o objeto pela última vez e entrar em lupe RASTREANDO todas as vezes que foi detectado um objeto em movimento.

O último estado, representado por VAGANDO tem um comportamento basicamente reativo e tem por função explorar o ambiente desconhecido simplesmente desviando-se de obstáculos (sejam eles fixos ou móveis) através da função *DesviaObstáculos*. A função computa as velocidades lineares e de rotação por meio de uma função da distância aos obstáculos sensoriados.

Finalmente, as referências de velocidade de rotação e linear são então aplicadas ao robô *Pioneer* através da função InputPioneer, que comunica-se diretamente com o sistema operacional do robô.

4.1.1 Desenvolvimento do ambiente

O primeiro passo para o desenvolvimento do ambiente é criar um arquivo com a extensão .cfg. Esse tipo de arquivo gera um mapa vazio, no qual posteriormente se configura as características essenciais do projeto. Algumas dessas características são configuras da seguinte forma :

```
{
    boundary 1; // Se existe ou não uma caixa delimitadora em torno do modelo
    gui_nose 0; //Indica qual é a regial frontal do robô
    gui_grid 0; //Cria uma grade de referência
    gui_move 0; //Possibilita mover o modelo para qualquer região do mapa
    gui_outline 0; //Indica se o modelo po ser descrito
    gripper_return 0;//habilita os parametros do retorno do LASER
    fiducial_return 0; //Indica o retorno do LASER para o modelo
    laser_return 1;// ativa o retorno do LASER
}
```

Esse arquivo mapa nada mais é que uma representação binária do ambiente, pois o computador só lê sinais elétricos, sem corrente e com corrente, representados respectivamente pelos números 0 e 1. Nesse caso 0 é espaço livre no mapa em que o robô vai ser inserido e 1 para obstáculos obstáculo.

O Algoritmo 2 descreve a interação do que existe entre o *LASER SICK* LMS200, o *Robô Pionner P3 - AT* e o mapa.

Algoritmo 4.2 Algoritimo de desenvolvimento do mapa

```
1: #include "pioneer.inc"
2: #include "map.inc"
3: #include "sick.inc"
4: # time to pause
5: paused 1
6: resolution 0.02
7: # configure the GUI
8: size [ 635.000 666.000 ]
9: # in pixels scale 37.481
10: # pixels per meter center [ -0.019 -0.282 ] rotate [ 0 0 ] show_data 1
12: # load an environment bitmap
13: floorplan (
14: name "cave"
15: size [16.000 16.000 0.800]
16: pose [0 0 0 0]
17: bitmap "bitmaps/cave.png" )
18: pioneerp3atdx (
19: "r0" pose [ -7 -7 0 45 ]
20sicklaser (
21: # ctrl "lasernoise"
22: #ctrl "wander"
23: # report error-free position in world coordinates localization "gps" localization_origin
[ 0 0 0 0 ] )
24: (
25: color "green"
26: pose [ 6.000 6.000 0 0 ]
27: name "pessoa" ctrl "sink
28: )
```

Nesta parte do algoritmo são definidas as interações que serão feitas pelo *Pioneer, LASER* e o mapa, sendo que este fica caracterizado na linha 2 e na linda 14. O *Layout* do mapa é importado no algoritmo pelo comando na linha 14, onde o *Stage* faz a leitura da planta baixa de qualquer ambiente exportado para o simulador e salvo em formato de figura .png. O ambiente gerado é tridimensional que será analisado somente um plano, pois o sensor *LASER SICK LMS200* faz a leitura bidimensional do ambiente captado.

Nas linhas de 23 a 28 do algoritmo de desenvolvimento do mapa é inserido o objeto móvel, que poderá ser movido para qualquer região do mapa.

5 Resultados

A simulação proposta visa que o robô *Pionner* dotado do *LASER SICK LMS200* cumpra o objetivo de rastrear objetos em movimento em um ambiente desconhecido, sempre desviando de obstáculos fixos, seguindo todos os protocolos estabelecidos pelo algoritmo.

5.1 Restrições do problema

Durante a etapa de validação do algoritmo foram impostas restrições ao problema de perseguição com o objetivo de contornar eventuais complicações na aplicação do algoritmo em uma plataforma real.

Os erros associados ao modelo matemático sempre são realçados em uma aplicação real. O sensor utilizado, embora disponha de precisão milimétrica, pode gerar erros que comprometam o desempenho do algoritmo em uma aplicação no robô *Pioneer*. Portanto, ainda na função *SubtrBckGnd*, caso a rotina para subtração de *background* retorne diferença entre a comparação de sensoriamentos, é aplicado então um filtro para assegurar que a subtração reflita, de fato, a presença de um objeto dinâmico no ambiente. Este filtro baseia-se em uma restrição imposta posteriormente ao projeto, justamente para atender esta necessidade de excluir os ruídos inerentes ao sensor durante o processamento.

A restrição consiste de uma uniformidade e continuidade do objeto a ser detectado. Em outras palavras, o objeto a ser rastreado deve representar para o sensor uma barreira contínua, livre de vãos, garantindo dessa maneira uma assinatura do objeto em si. Isso leva a outra restrição do problema que pode não ter ficado explícita durante a apresentação do problema: O robô é capaz de seguir um único objeto móvel. Uma terceira restrição é decorrente das duas anteriores e, segundo a arquitetura do robô e confecção de seu algoritmo, revela que: Em um cenário onde há a presença de mais de um objeto móvel, o robô ignora os demais e persegue apenas aquele que está mais a sua esquerda.

5.2 Simulação no *Player/Stage*

A validação do algoritmo se deu através de etapas baseadas nos passos da máquina de estado e foram avaliadas conforme as figuras 30 a 34.

Na primeira etapa da simulação inicia-se chamando o ambiente, como ilustra a imagem 30, em qual o robô vai ser inserido. Nesta etapa alguns parâmetros são estabelecidos pelas funções *ConfigParam* e *IniciaParam*, conforme o pseudocódigo no Algoritmo 1. Tais como: o posicionamento e direção do robô, rotação em torno do seu próprio eixo

igual a zero, quantidade de feixes do *LASER* necessárias para identificar a detecção de movimento e algumas outras constantes que servem para indicar alguma anomalia no ambiente, garantindo desta forma que o robô tenha uma velocidade inicial igual a zero e faz uma preparação para receber a primeira etapa da máquina de estado.

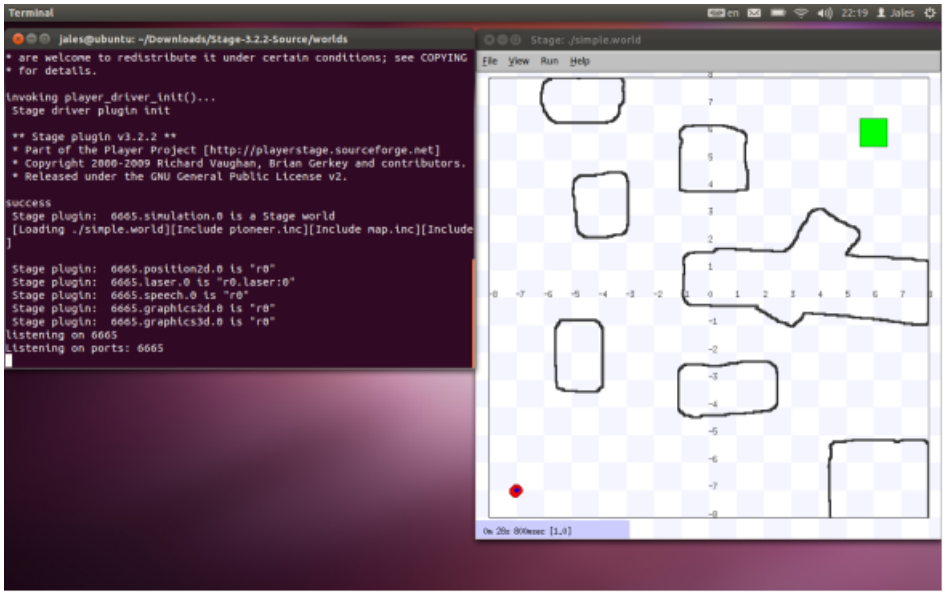

Figura 30 – Ativação do mapa em que o robô vai esta inserida

Fonte: (Próprio autor)

Com a ativação do mapa inicia-se o primeiro modo da máquina de estado denominada de ESCANEANDO, figura 31. É escolhido este estado para ser inicial, pois o robô desconheceu o ambiente, garantindo dessa forma que o robô vai ter uma velocidade inicial igual a zero. Logo em seguida o algoritmo vai recebendo os valores das leituras dos feixes e vai comparando com as leituras subsequentes. Com os dados interpretados é possível que o robô possa iniciar sua trajetória sem que venha colidir com obstáculos. Nessa etapa caso não seja identificado algum objeto em movimento o robô entra no modo VAGANDO. Uma característica do algoritmo é que enquanto o robô está em movimento ele não faz o escaneamento do ambiente, apenas faz a trajetória de desvio de obstáculos.

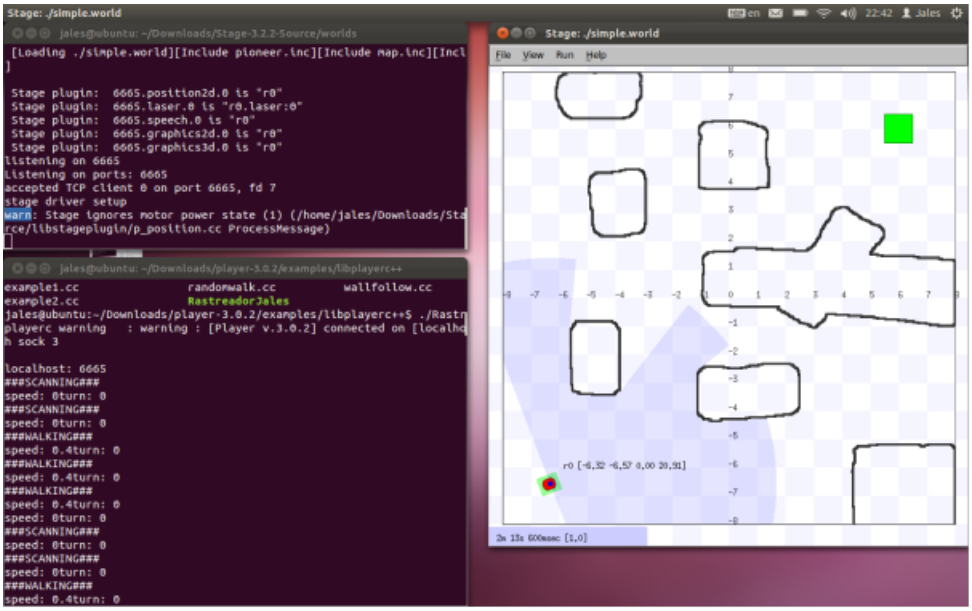

Figura 31 – Início da simulação tendo o estado inicial de escaneamento

Fonte: (Próprio autor)

Na identificação de um objeto em movimento pela técnica de subtração de fundo, inicia-se o modo RASTREANDO, figura 32. Nessa etapa o robô faz uma rotina de alinhamento com as referências do objeto em movimento, direcionando ao ponto entre a primeira leitura, que fica estabelecida como ponto inicial e a última leitura estabelecida como ponto final.

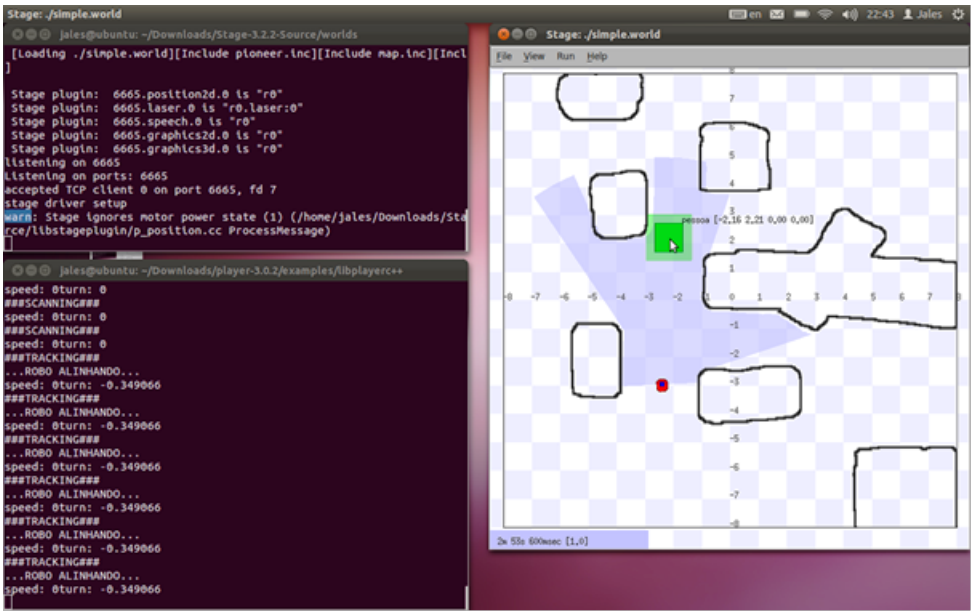

Figura 32 – O modo de rastreamento e alinhamento em direção ao alvo

Fonte: (Próprio autor)

Com o robô alinhado e ainda no modo RASTREANDO, figura 33, o algoritmo inicia-

se a rotina de perseguição entrando em rota de colisão, nesta rotina o algoritmo ativa o comportamento reativo para que o robô não venha sofrer algum dano, caso o ambiente sofra alguma mudança repentina. Desta forma fica garantido que o robô irá fazer a perseguição do objeto e entrar no modo de ESCANEANDO todas as vezes que atingir o ponto final da trajetória estabelecida pelo algoritmo.

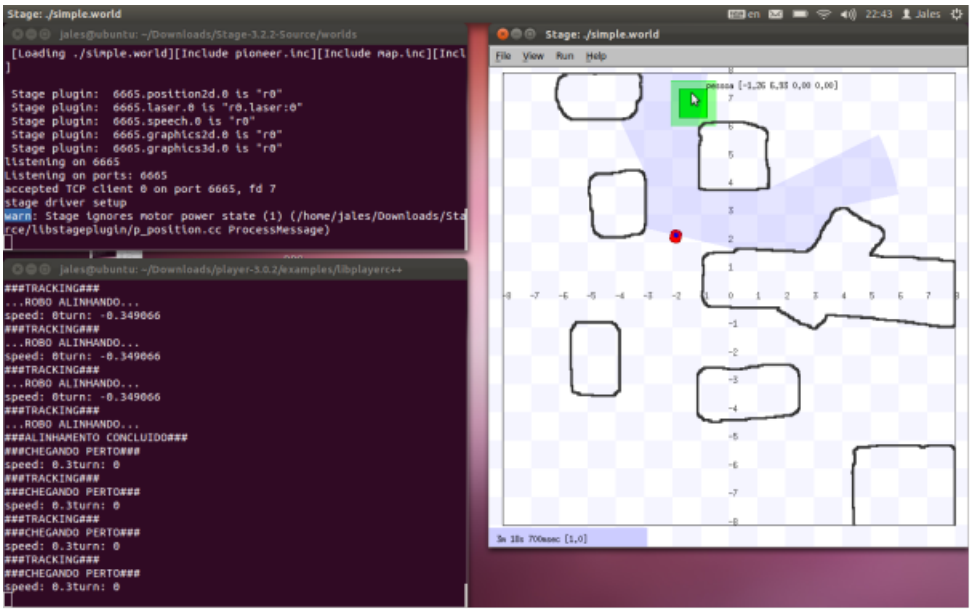

Figura 33 – Alinhamento concluído e aproximação do alvo em movimento

Fonte: (Próprio autor)

Com a aproximação limite estabelecida o algoritmo, como sendo de 50cm cumpre-se o objetivo e mantém uma perseguição, conforme ilustra a figura 34.

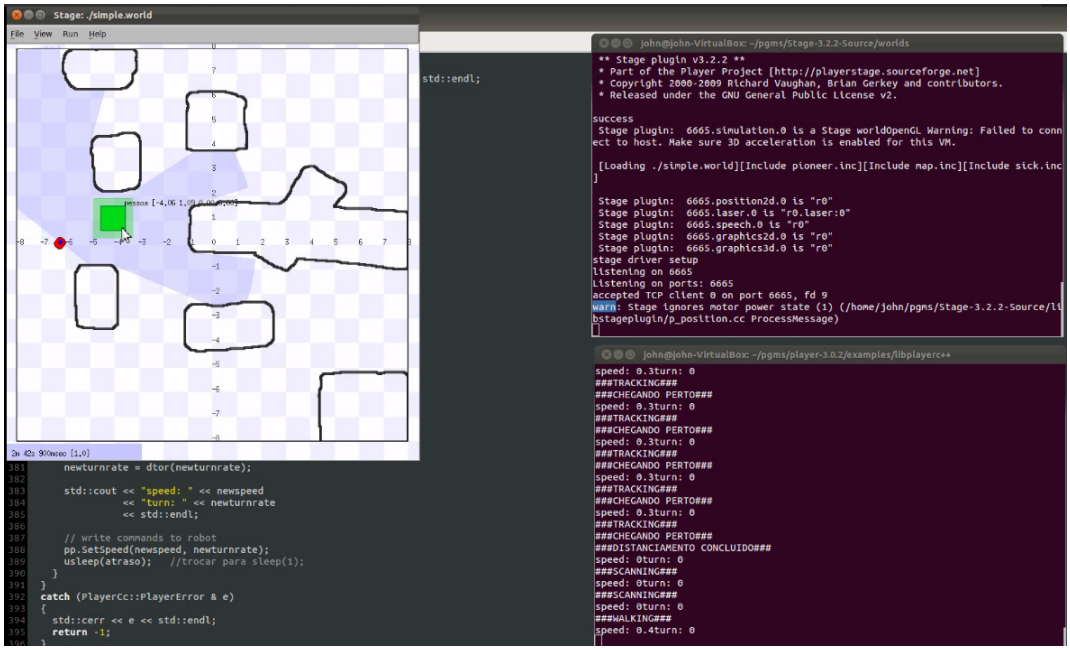

Figura 34 – Objetivo concluído

Fonte: (Próprio autor)

Porém durante e após a elaboração do algoritmo alguns problemas foram contornados que serão divididos segundo os tópicos para uma melhor avaliação.

5.2.1 Algoritmo de rastreamento

O algoritmo de rastreamento de objetos foi validado conforme as cinco etapas apresentadas na subseção. Contudo, em uma primeira tentativa houve resultados inesperados que tiveram que ser contornados, os mesmos é a seguir detalha

Na execução da etapa do Modo RASTREANDO, o algoritmo funcionou perfeitamente para valores que a média angular estimada de giro do robô no seu próprio eixo em direção ao alvo não ultrapassasse valores maiores que $+180^{o}$ ou menores -180^{o}, valores positivos quando o giro era no sentido anti-horário e valores negativos quando o giro tivesse no sentido horário. Caso esses valores ultrapassassem o algoritmo entraria em uma situação de perda de referência, em que o robô entraria em um giro infinito em seu próprio eixo.

Para contornar esse problema uma condição de ajuste foi imposta:

- Se o ângulo de ajuste ultrapassar valores acima de 180^{o} ou abaixo o algoritmo faz uma mudança do ângulo desejado, invertendo esses valores para outro plano, como exemplo:

$$\text{Desejado} = \text{Referência local } (\mathbf{100^{o}}) + \text{Referência Global } (\mathbf{81^{o}})$$

$$\text{Ajusta } (\mathbf{Desejado = 181^{o}})$$

enquanto(Alinhamento())

{

se ((**360º - 181º = 179º**) < limite)

Ajusta (**Desejado = 179º**)

}

Desta forma é enviando instruções ao robô que o seu giro angular não será um valor acima de 180º, garantido desta forma que o algoritmo não perca referência.

6 Conclusões e trabalhos futuros

Neste trabalho, foi exposta uma abordagem que trata da navegação autônoma de robôs móveis sem o conhecimento prévio do ambiente. Além disso, como objetivo secundário, foi configurado um ambiente totalmente desconhecido pelo algoritmo de navegação, sempre tendo como objetivo principal o desvio de obstáculos e o rastreamento de um objeto em movimento. O algoritmo desenvolvido foi baseado em uma máquina de estado composta de três estados. O estado inicial (ESCANEANDO) é responsável pelo reconhecimento do ambiente e a verificação de objetos dinâmicos, através da utilização da técnica de anomalia. Caso o algoritmo detecte alguma anomalia, o segundo modo da máquina de estado é comutado para o modo RASTREANDO. Neste estado são realizados os ajustes de rotação do eixo e velocidade do robô para que ele vá de encontro ao objeto a ser rastreado. O terceiro e último estado, denominado VAGAR, é chamado caso o algoritmo utilizado na subtração de fundo não acuse movimento no ambiente. O sensoriamento do ambiente foi feito através de um sensor *LASER SICK LMS200* de varredura instalado a bordo do robô móvel utilizado, o *PIONEER P3-AT*.

Com o objetivo de lidar com a detecção de objetos móveis e tratar informações provenientes do sensor *LASER* foi desenvolvido a técnica para a auto-localização, a fim de que o robô seja orientado no ambiente em relação a sua posição e orientação. Deste modo, o robô é capaz de tomar as decisões de planejar as ações, executar e determinar a trajetória do seu objetivo desejado.

A fim de se comprovar a funcionalidade dos conceitos-chaves, utilizados para o problema de rastreamento, foi escrito um projeto em linguagem C++, graças ao suporte dos softwares escolhidos para a validação do trabalho. Este programa admite como entrada os parâmetros intrínsecos ao funcionamento do robô. Desta forma, foram realizadas simulações com o auxílio dos *software Stage*, responsável pela edição dos ambientes onde o robô será posto para navegar, e o *software Player* que é o simulador propriamente dito, no qual no mesmo algoritmo leva-se em considerações os erros dos sensores presentes no robô.

A característica da arquitetura deliberativa fica evidente quando inicialmente o algoritmo faz uma verificação, procurando objetos em movimento e determinando uma trajetória inicial de movimentação. O que caracteriza uma arquitetura deliberativa, que tem como essência o planejamento das ações a serem tomadas e suas decisões. Por outro lado a arquitetura reativa fica evidente no algoritmo quando o robô responde a uma modificação inesperada do ambiente.

Foi optado pelo emprego das versões 3.0.2 para o software *Player* e 3.2.2 do *Stage*. Essa escolha deveu-se às estabilidades apresentadas por essas versões e relatadas como confiáveis em outros trabalhos relacionados.

As instalações do *Player* e *Stage* não são simples, porém exigiu alguns conhecimentos do sistema *Linux.* No entanto, uma das desvantagens está na configuração correta do ambiente que depende de muitas dependências que variam com a versão do *software* combinado com a distribuição do *Linux.*

Durante a execução da técnica da subtração de *background* erros de leitura devido ao suporte de incorporeamento de falhas dos *software* foram algumas vezes detectados. Como o sinal que o algoritmo, é suscetível a ruídos em determinados feixes do sensor, fez-se necessária a eliminação dessas falsas detecções de obstáculos. Portanto foi projetado um filtro que remediou este ponto fraco e tornou conseguinte o algoritmo mais robusto a falhas.

Foi observado, que durante a aplicação desse filtro também ficou resolvido o problema da presença de vários objetos em movimento. Posteriormente, o problema foi restringido a somente um objeto para evitar falhas inesperadas na perseguição do alvo.

A estrutura necessária para que o código funcionasse não possibilitava *loops* extensos. Após constatar isso foi mudada a estrutura do código de forma a adotar *flags* de sinalização para entrar nos estados e etapas específicas desses estados.

Do ponto de vista do aprendizado, o trabalho foi extremamente enriquecedor no que diz respeito, a saber, gerenciar o tempo de desenvolvimento de um projeto e a manter a motivação mesmo diante de dificuldades no desenvolvimento, além de ter contribuído para novos aprendizados e amadurecimento de conhecimentos.

6.1 Trabalhos futuros

Pretende-se futuramente realizar diversos trabalhos como continuidade do apresentado neste trabalho. Algumas propostas a serem consideradas são relacionadas a seguir:

- Embarcar o programa na plataforma do robô *Pionner P3-AT*;
- Resolver o problema da subtração de *background* para obstáculos não contínuos, pois durante a elaboração do algoritmo restringiu-se a objetos com o corpo ininterruptos;
- Publicações em congressos científicos;
- Estudar o benefício do emprego de outros sensores.;
- Estudar a técnica de fusão sensorial para adicionar outros sensores e poder fazer uma compreensão mais completa do ambiente, a fim de lidar com uma possibilidade maior de obstáculos presentes em diferentes terrenos;
- Estudar uma técnica para o rastreamento de um objeto não somente quando o robô estiver com velocidade nula, mas quando estiver se locomovendo;

- Desenvolver outras finalidades para o robô, assim como procurar outras plataformas similares presentes no mercado e com suporte a *softwares* livres de desenvolvimento em robótica.

Referências

BAJRACHARYA, M. M. Gamma-slam: Using stereo vision and variance grid maps for slam in unstructured environments. v. 19, n. 10.1109, p. 3717 – 3724, 2008. Citado na página 18.

BRäUNL, T. *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems*. [S.l.: s.n.], 2008. 150 p. Citado na página 27.

COLARES, R.; CHAIMOWICZ, L. Planning for simultaneous localization and mapping using topological information. v. 16, n. 10.1109, p. 214 – 219, Outubro 2012. Citado na página 14.

CUTKOSKY, M. R. *Human Grasp Choice and Robotic Grasp Analysis*. [S.l.: s.n.], 1990. 5 - 31 p. Citado na página 33.

ELFES, A. Sonar-based real world mapping and navigation. p. 249 – 265, 1987. Citado na página 41.

EVERETT. 2014. Disponível em: <http://www.public.navy.mil/spawar/Pacific/Robotics/Pages/Comms_Overview.aspx>. Citado na página 20.

FERREIRA, F. Robô de resgate deselvolvido no modelo do desafio da robocup junior. p. 1– 3, 2004. Citado na página 41.

FREIRE, E. O. *Desenvolvimento de um Sistema de Sensoriamento Ultra-Sônico Para um Robô Móvel com Controle Baseado em Agentes*. Dissertação (Dissetação de mestrado) — Universidade Federal do Espírito Santo, Vitória - ES, 1997. Citado na página 13.

GRASSI, V. *Arquitetura Híbrida para Rôbos Móveis Baseada em Função de Navegação com Intereração Humana*. Dissertação (Tese de Doutorado) — Universidade de São paulo, São Paulo, 2006. Citado 3 vezes nas páginas 37, 38 e 39.

HEINEN, F. J. *Sistema de Controle Híbrido para Robôs Móveis Autônomos*. Dissertação (Dissertação de mestrado) — Universidade do Vale do Rio dos Sinos, São Leopoldo - RS, 2002. Citado 3 vezes nas páginas 14, 22 e 23.

JUNG, C. R. Computação embarcada: Projeto e implementação de veículos autônomos inteligentes. v. 10, n. 2, p. 1358 – 1406, Junho 2005. Citado na página 18.

KLIPP, T. dos S. *Proposta de uma arquitetura para alocação de tarefas em grupos de robôs móveisbaseado em acordo bizantino*. Dissertação (Dissertação) — Universidade Federal de Santa Catarina, Florianópolis - SC, 2013. Citado na página 13.

KUO, C. H.; SYU, Y. S. An embedded robotic wheelchair control architecture with reactive navigations video. n. 2161-8070, p. 810–815, Agosto 2011. Citado na página 41.

LEI, B.; LI, W. A fuzzy behaviours fusion algorithm for mobile robot real-time path planning in unknown environment. n. 10.1109, p. 173–178, Março 2007. Citado na página 40.

LUCA, C. Trajectory tracking control os a four-wheel, differentially driven mobile robot. In: *IEEE, International Conference on Robotic & Automation*. [S.l.: s.n.], 1999. Citado na página 28.

MARQUES, A. R. *Computação Reconfigurável Aplicada á Robótica (FPGA)*. 2014. Disponível em: <http://www2.eletronica.org/artigos/eletronicadigital/computacao-reconfiguravel-aplicada-a-robotica-fpga>>. Citado na página 13.

NEHMZOW. Complex robot training tasks through bootstrapping system identification. v. 14, n. 10629420, p. 2168 – 2173, Fevereiro 2009. Citado na página 18.

NERIS. *Um piloto automático para aeronaves do projeto ARARA*. Dissertação (Dissestação de mestrado) — Universidade de São paulo, 2001. Citado na página 19.

PATROLBOT. *MobileRobots Inc.* 2014. Disponível em: <http://www.mediander.com/connects/2150115/patrolbot/#!/>. Citado na página 13.

PEDROSA, D. Uma prpostavde slam com determinação de informações geometricas do ambiente. *DCA*, v. 12, n. 59078 - 900, p. 1704 – 1709, 2013. Citado na página 14.

PIERI, E. R. de. *Curso de Robótica Móvel*. 2002. Citado 2 vezes nas páginas 26 e 38.

ROBOTS, M. *Autonomous mobile robot cores, bases and accessories*. 2014. Disponível em: <http://www.mobilerobots.com>. Citado na página 15.

ROOMBA, I. 2014. Disponível em: <https://www.irobot.com.br/>. Citado na página 18.

ROSáRIO, J. *Controle e Automação*. [S.l.: s.n.], 2008. 250 p. (2008, 85-7605-010-2). Citado na página 18.

SIEGWART, R. *Introduction to Autonomous Mobile Robots*. 2004. Citado na página 27.

VIEIRA, F. C. *Controlador Adaptativo Robusto para um Robô Móvel com acionamento diferencial*. Dissertação (Dissertação) — Universidade Federal do Rio Grande do Norte, Rio Grande do Norte - RN, 2005. Citado 2 vezes nas páginas 19 e 20.

ZEKI, S. *Vision of the Brain*. [S.l.: s.n.], 1993. Citado na página 33.

APÊNDICE A – Algoritmo para perseguição de objetos em movimento

/* Algoritmo para perseguição de objetos em movimento */

```
//Aqui foram algumas bibliotecas necessarias
#include <libplayerc++/playerc++.h>
#include <iostream>
#include <unistd.h>
#include <stdlib.h>
#include "args.h"
//Aqui embaixo tem alguns DEFINES
#define RAYS 32
#define ATRASO_PADRAO 1000000/2
#define SCANNING 0 //Escaneia o ambiente.
#define TRACKING 1
#define VAGANDO 2 //Vaga sem rumo, por um tempo definido em TEMPO_VAGAR
#define TEMPO_ESPERA 2
#define TEMPO_VAGAR 3
#define TAM_MINIMO 2
#define TOL_MAX 1.05
#define TOL_MIN 0.95
#define QUAR_GRAUS 40
#define VINT_GRAUS 20
#define VEL_MINIMA 5
//Essas linhas são responsaveis por dar um start inicial
   static int it = 0;
   static int estado = SCANNING; //Define o estado inicial como sendo SCANNING!!!
int main (int argc, char **argv)
{
   parse_args(argc,argv);
//Aqui embaixo são as declaracao de variaveis, em alguns casos já sao carregados alguns valores
   double newspeed = 0;
   double newturnrate = 0;
   double l,r,minR,minL;
```

```
    double deslinicio,deslfim;
    double vel_rot=0;
    double desejado;
    double err_ok = 0.05236;
    double fator=1432;
    int desl[360]; int qty = 0;
    int flagAlinhamento=0;
    int flagDistanciamento=0;
    int flagDesvio = 0;
    int anginicio,angfim,angtrack;
    static long int atraso = ATRASO_PADRAO;
    static double ran[360];
    static double distinicio;
    static double distfim;
    static double ran_temp[360];
    static double teste[360];
    double angrobot;
try
{
using namespace PlayerCc;
//Aqui são as inicializacões padrão do robô e do LASER. Atribuindo nome ao robô,
ao LASER e liga motores
    PlayerClient robot(gHostname, gPort);
    Position2dProxy pp(&robot, gIndex);
    LaserProxy lp(&robot, gIndex);
    std::cout << robot << std::endl;
    pp.SetMotorEnable (true);
for(;;)
{
robot.Read();
    angrobot = pp.GetYaw();
    minR = lp.GetMinRight();
    minL = lp.GetMinLeft();
    if(estado == SCANNING)
{
        std::cout << "###SCANNING###" << std::endl;
        r = 0; l = 0;
        vel_rot = 0;
        if(it<1);
```

```
        {
        if(it<1)
    {
        for(int i=0;i<360;i++) { ran_temp[i] = lp.GetRange(i);
    }
        else { for(int i=0;i<360;i++) { ran[i] = lp.GetRange(i); // armazena a primeira
leitura para a comparação
    }
        qty = 0;
        for(int i=0;i<360;i++)
    {
        if(ran[i]==ran_temp[i])
    {
        desl[i]=0;
    }
        else
    {
        desl[i]=1;
        qty = qty + 1;
    }
        it = it+1;
        if(qty>TAM_MINIMO)
    {
        int j=0;
        int count = qty;
        while((count>0)&&(j<360))
    {
        if(desl[j]==1)
    {
        if(count==qty)
    {
        anginicio = j;
        distinicio = ran[j];
        deslinicio = ran_temp[j]-ran[j];
    }
        count = count - 1;
        if(count==0)
    {
        angfim = j;
```

```
        distfim = ran[j];
        deslfim = ran_temp[j]-ran[j];
        angtrack = (angfim + anginicio)/2;
        angrobot = pp.GetYaw();
        desejado = angrobot + lp.GetBearing(angtrack);
        if(desejado>3.1415)
        desejado = -1*(6.2431 - desejado);
        if(desejado<-3.1415)
        desejado = 6.2431 + desejado;
}
        j=j+1;
}
        qty = 0;
        it = 0;
        estado = TRACKING;
        flagAlinhamento=1;
}
        else
{
        if(it>=TEMPO_ESPERA)
{
        it=0; qty = 0;
        estado = VAGANDO;
}
        else if(it>=2)
{
        for(int i=0;i<360;i++)
        ran_temp[i] = ran[i];
}
        else if(estado == TRACKING) //faz rotina de perseguição
{
        std::cout << "###TRACKING###" << std::endl;
        if(flagAlinhamento)
{
        vel_rot = 20;
        std::cout << "...ROBO ALINHANDO..." << std::endl;
        if(abs(angrobot-desejado)>2.00)
{
        if((angrobot<-1.5707)&&(desejado>1.5707))
```

```
{
        angrobot = angrobot+6.2832;
}
        else if((angrobot>1.5707)&&(desejado<-1.5707))
{
        angrobot = angrobot - 6.2832; }
}
        if(angrobot>(desejado+err_ok))
{
        vel_rot = -1*vel_rot;
        atraso = int(1000*fator*(abs(desejado-angrobot)));
}
        else if(angrobot<(desejado-err_ok)) //Objeto a esquerda do robÃ´l
{
        atraso = int(1000*fator*(abs(desejado-angrobot)));
}
else
{
        flagAlinhamento=0; flagDistanciamento=1;
        vel_rot = 0;
        std::cout << "###ALINHAMENTO CONCLUIDO###" << std::endl; //Aconteceu alguma virada de referência
        atraso = ATRASO_PADRAO;
}
        if(flagDistanciamento)
{
//Vai atrás do alvo se tiver a mais de 50cm dele = distância de segurança
        std::cout << "###CHEGANDO PERTO###" << std::endl;
        if(deslfim>0)
{
        if(distfim>0.5)
{
        l = (1e5*minR)/500-100;
        r = (1e5*minL)/500-100;
        if (r > 100) r = 100;
        vel_rot = r-l;
        flagDesvio = 1;
}
        if((minL>=0.50)&&(minR>=0.50))
```

```
{
    if(flagDesvio)
{
    l=0;
    r=0;
    vel_rot=0;
    flagDesvio = 0;
    flagDistanciamento=0;
    estado = SCANNING;
}
    else { distfim = distfim - 0.7;
    l=75;
    r=75;
}
    else
{
    l=0;
    r=0;
    flagDistanciamento=0;
    estado = SCANNING;
    std::cout << "###DISTANCIAMENTO CONCLUIDO###" << std::endl;
}
    else
{
    if(distinicio>0.5) //Vai atrás do alvo se tiver a mais de 50cm dele
{
    l=0;
    r=0;
    if((minR<0.45)||(minL<0.45))
{
    l = (1e5*minR)/500-100;
    r = (1e5*minL)/500-100;
    if (l > 100)
    l = 100;
    if (r > 100)
    r = 100;
    vel_rot = r-l;
    flagDesvio = 1;
}
```

```
        if((minL>=0.45)&&(minR>=0.45))
{
        if(flagDesvio)
{
        l=0;
        r=0;
        vel_rot=0;
        flagDesvio = 0;
        flagDistanciamento=0;
        estado = SCANNING;
}

        else
{
        distinicio = distinicio - 0.7;
        l=75;
        r=75;
}
        else
{
        l=0;
        r=0;
        flagDistanciamento=0;
        estado = SCANNING;
        std::cout << "###DISTANCIAMENTO CONCLUIDO###" << std::endl;
}
        else if(estado == VAGANDO) //faz rotina pra ficar vagando
{
        if(it<TEMPO_VAGAR)
{
        std::cout << "###WALKING###" << std::endl;
        it = it+1;
        l = (1e5*minR)/500-100;
        r = (1e5*minL)/500-100;
        if (l > 100) l = 100;
        if (r > 100) r = 100;
}
        else
{
```

```
        it=0;
        estado = SCANNING;
        r = 0;
        l = 0;
}
        vel_rot = r-l;
}
        newspeed = (r+l)/5e2;
        newturnrate = vel_rot;
        newturnrate = limit(newturnrate, -40.0, 40.0);
        newturnrate = dtor(newturnrate);
        std::cout << "speed: " << newspeed << "turn: " << newturnrate << std::endl;
        pp.SetSpeed(newspeed, newturnrate);
        usleep(atraso);
}
        catch (PlayerCc::PlayerError & e)
{
        std::cerr << e << std::endl; return -1;
}
```