Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística

**Implementação do Método de Campos
Potenciais para Navegação de Robôs Móveis
Baseada em Computação Reconfigurável.**

Rovilson Mezencio

Orientação: Prof. Dr. Eduardo Marques

Monografia apresentada ao Instituto de Ciências Matemáticas e de Computação - USP, para o Exame, como parte dos requisitos para a obtenção do título de Mestre em Ciências - Área de Ciências de Computação e Matemática Computacional.

USP - São Carlos

AGOSTO DE 2002

Dedico este trabalho à pessoa que tem sido motivo do meu crescimento: Minha esposa, Cejana.
Ela não faz idéia de quanto devo a ela.
Espero, entretanto, que ela me permita ficar pagando por toda a vida.

Agradecimentos

Muitas pessoas ao meu redor contribuíram para que este trabalho pudesse ser realizado ao longo de quase três anos. Quando falo em contribuição não me refiro apenas a parte técnica mas sim ao suporte moral e emocional. A vida profissional, financeira e familiar tiveram que ser conciliadas com toda a carga que mestrado traz. Sem as pessoas, a seguir, tal sonho não teria se realizado. Agradeço:

Ao Prof. Dr. José Pacheco de Almeida Prado, que como um pai, me conduziu nos primeiros passos da carreira científica e fez com que eu acreditasse que eu podia chegar, quando nem eu mesmo tinha certeza.

Ao Prof. Dr. Eduardo Marques que me adotou nesta nova casa e sem medir nenhum esforço me acompanhou com atenção, afeto e paciência ilimitada ainda nos momentos em que não fazia por merecer. O prof. Eduardo foi muito, sem dúvida nenhuma, muito além da condição de mero orientador.

A Profa. Dra. Roseli Francelin Romero por ter colocado seus conhecimentos a nossa disposição e pela iniciativa e direção do projeto ARMOSH sem o qual nada disto seria possível.

Ao Dr. Percival Xavier Rebelo Filho e Sra. Maria Helena Leal Lúcio Rebelo, pais de minha esposa, que acompanharam de perto o percorrer deste caminho sempre estimulando e apoiando quando se fazia necessário.

A minha querida esposa, Cejana, pela paciência infindável, companheirismo, carinho, motivação e por existir. Sem você nada disto seria possível.

Por fim, porém não menos importante, aos meus pais José e Maria e meus irmãos Ricardo e Adriana, os quais possibilitaram que todas as outras pessoas, anteriormente citadas, se relacionassem comigo de forma harmônica e respeitosa, graças a educação que me incutiram no decorrer da vida.

Do ponto de vista técnico, ainda, outras pessoas se fizeram importantes. Contribuições sem as quais este trabalho não teria brilho ou até, não teria se findado, são elas: Olle Wijk, da Kungl Tekniska Hogskolan – Estocolmo – Suécia, porque apesar da sua posição, conhecimento e distância não deixou em momento algum de me ceder o material e as explicações os quais eu necessitava.

Sérgio Gonsalves Tourino ,aluno da Universidade de Brasília, o qual gentil e crucialmente colaborou no momento em que já não tínhamos muitos caminhos. Foi ele quem nos cedeu o código fonte de navegação tão esperado. Mesmo sendo eu um perfeito estranho, fui tratado como um velho amigo.

Leandro Henrique Mendonça de Oliveira porque apesar do cotidiano atribulado em que estávamos todos envolvidos em Goiânia ele, estando as vésperas de seu casamento, conseguiu tempo para corrigir e me ajudar na escrita desta dissertação.

Ao Murilo Inácio Rodrigues pelas longas horas de trabalho no Laboratório de Computação Reconfigurável da USP (São Carlos). A contribuição técnica trazida pelo seu trabalho do mestrado foi conclusiva para o término deste trabalho. Apesar de pouco conhecê-lo nos dias em que convivemos juntos, no LCR, se mostrou cordial e companheiro como um irmão.

Aos colegas do LCR: Denis, Marco Antônio, Alexandre, Jean, Richard e Leandro Carlos por nunca se omitirem a um apoio técnico e principalmente moral.

A Deus pelo apoio técnico, moral e espiritual e por me permitir não só concluir o trabalho, mas também ter me relacionado com todas essas pessoas.

Índice Remissivo

CAPÍTULO 1 – INTRODUÇÃO	1
1.1 MOTIVAÇÃO	1
1.2 JUSTIFICATIVA	1
1.3 OBJETIVO	2
1.4 DELIMITAÇÃO	2
1.5 APRESENTAÇÃO	3
CAPÍTULO 2 – ROBÔS MÓVEIS	5
CAPÍTULO 3 – FPGAS E TRABALHOS RELACIONADOS	14
3.1 A COMPUTAÇÃO RECONFIGURÁVEL	15
3.2 RECONFIGURAÇÃO DINÂMICA	17
3.3 SISTEMAS DE COMPUTAÇÃO RECONFIGURÁVEL	18
3.4 O KIT DE DESENVOLVIMENTO EXCALIBUR	19
3.5 O PROCESSADOR NIOS	19
3.6 COMPILADOR GNUPRO	21
3.7 QUARTUS II ® – FERRAMENTA DE DESENVOLVIMENTO	21
3.8 PLACA DE DESENVOLVIMENTO	22
3.9 PARADIGMA SOC	22
4.0 TRABALHOS RELACIONADOS	24
CAPÍTULO 4 – NAVEGAÇÃO DE ROBÔS MÓVEIS	26
4.1 DOMÍNIO FÍSICO	26
4.2 DOMÍNIO LÓGICO	26
4.2.1 Posicionamento odométrico	27
4.2.2 Localização Global Dentro de um Mapa	27
4.2.3 Construção de Mapas e localização simultânea	27
4.3 OS PARADIGMAS	28
4.3.1 Paradigma baseado em Células – Geométrico	28
4.3.2 Paradigma topológico	29
4.3.3 Paradigmas Híbridos	30
4.4 RESUMO	33
CAPÍTULO 5 – OS TIPO DE MAPAS DE CÉLULAS	34
5.1 DESEMPENHO NA SALA	35
5.2 DESEMPENHO NO CORREDOR	37
5.3 RESUMO E DISCUSSÕES	40
5.4 HIMM X CAMPOS POTENCIAIS. QUAL A RELAÇÃO?	41
5.5 A HIERARQUIA DAS TAREFAS.	42
5.6 POR QUE CAMPOS POTENCIAIS E NÃO HIMM ?	43
CAPÍTULO 6 – CAMPOS POTENCIAIS	45
6.1 O MÉTODO CAMPOS POTENCIAIS NO MUNDO.	48
6.2 A OBTENÇÃO DO CÓDIGO.	52
6.3 O ESTUDO DE UM CASO.	52
6.4 O ESTUDO DOS MÓDULOS.	54
6.5 O ESTUDO DO CÓDIGO.	57
6.7 Calibração de variáveis	59
CAPÍTULO 7 – IMPLEMENTAÇÃO DO ALGORITMO DE CAMPOS POTENCIAS EM HARDWARE RECONFIGURÁVEL	64
7.3 HARDWARE DEDICADO	65
7.3.1 As Funções Trigonométricas	67
7.3.2 Aritmética de Ponto Flutuante	68
7.3.3 O Teste dos Polinômios	71
7.5 O EXEMPLO IMPLEMENTADO	73
7.6 AS MEDIÇÕES	74
CAPÍTULO 8 – CONCLUSÃO E TRABALHOS FUTUROS	78
8.1 TRABALHOS FUTUROS	78
APÊNDICE A	87
CÓDIGOS FONTES E DADOS (TEXTOS) UTILIZADOS EM MATLAB	87

Índice de Figuras

FIGURA 1 – ÁRVORES DE ESCOLHAS DESTE TRABALHO.....	3
FIGURA 2 – CLASSIFICAÇÃO DOS ROBÔS QUANTO À PARTE FÍSICA	6
FIGURA 3 – EXEMPLO DE VEICULO PLANETÁRIO	7
FIGURA 4 - A) ROBÔ ENFERMEIRO DE BORENSTEIN B) ROBÔ ELDERY, CODINOME <i>FLO</i> DE UM GRUPO DA CMU	8
FIGURA 5 - A) FOTO REAL DO NAVBELT B) FIGURA EXPLICATIVA DO NAVBELT	9
FIGURA 6 - FOTO REAL DO NAVCHAIR	10
FIGURA 7 - A) VISÃO TOTAL DA LATERAL DO HERMES B) SEQUÊNCIA DE MOVIMENTOS ONDE HERMES SERVE UMA PESSOA.	11
FIGURA 8 - A) FOTO DA BASE KOALA, VENDIDA COMERCIALMENTE B) FIGURA VACCUM CLEAR.....	12
FIGURA 9 – A) RHINO B) MINERVA EM TRÊS ÂNGULOS.....	13
FIGURA 10 - ESTRUTURA GENÉRICA DE UM DISPOSITIVO FPGA.....	14
FIGURA 11 - ESQUEMA DE RECONFIGURAÇÃO	17
FIGURA 13- DIAGRAMA DE BLOCOS DO PROCESSADOR NIOS®	20
FIGURA 12 - MÚLTIPLOS NIOS EM UM APEX.....	21
FIGURA 14 - REFERÊNCIA DE PROJETOS DE PERIFÉRICOS SOPC	23
A) MAPA MÉTRICO B) DIAGRAMA DE VORONOI C) PONTOS CRÍTICOS D) LINHAS CRÍTICAS E) REGIÕES TOPOLÓGICAS F) GRAFO TOPOLÓGICO.....	31
FIGURA 15 - EXTRAÇÃO DE MAPAS TOPOLÓGICOS SOBRE MAPAS BASEADO EM GRIDS	31
FIGURA 16 - MAPA BASEADO EM GRID CONSTRUÍDO EM PARALELO AO TOPOLÓGICO.....	32
FIGURA 17 – EXTRAÇÃO DE UM MAPA TOPOLÓGICO DA MAPA DE CÉLULAS.....	33
FIGURA 18 – A) VISTA SUPERIOR DA SALA B) VISTA SUPERIOR DO CORREDOR. EM AMBOS OS CASOS A LINHA REPRESENTA A TRAJETÓRIA DO ROBÔ	34
FIGURA 19 - A) DADOS DO SONAR USADOS NO EXPERIMENTO B) MAPA TBF C) BAYESIANO D) DEMPSTER E) FUZZY F) BORENSTEIN.....	37
FIGURA 20. A) RASTREAMENTO REALIZADO PELO SONAR NO EXPERIMENTO CORREDOR B) MODELAGEM OBTIDA PELO MÉTODO NO TBF NO CORREDOR. C) MODELAGEM OBTIDA PELO MÉTODO BAYESIANO NO CORREDOR.....	38
FIGURA 21 - A) MODELAGEM OBTIDA PELO MÉTODO NO DEMPSTER-SHAFFER NO CORREDOR. B) MODELAGEM OBTIDA PELO MÉTODO FUZZY NO CORREDOR. C) MODELAGEM OBTIDA PELO MÉTODO DE BORENSTEIN NO CORREDOR.....	39
FIGURA 22 – ÁRVORE GENEALÓGICA DA RELAÇÃO ENTRE HIMM E CAMPOS POTENCIAIS.	42
FIGURA 23 – RELAÇÃO ENTRE OS TIPOS DE TAREFAS REALIZADAS POR ROBÔS MÓVEIS	43
FIGURA 24 – ESTE PROJETO DENTRO DO CONTEXTO TRADICIONAL	45
FIGURA 25 – TIPOS DE CAMPOS POTENCIAIS	46
FIGURA 26 – PIONEER DO LTI - PCS/EPUSP	49
FIGURA 27 – CAMPOS POTENCIAIS UTILIZADOS EM GRUPOS DE ROBÔS.....	51
FIGURA 28 - DIAGRAMA DE COMUNICAÇÃO DOS MÓDULOS EM MATLAB.....	55
FIGURA 29 – TESTES REALIZADOS COM OS MÓDULOS	56
FIGURA 30 – ESTUDO DOS MÓDULOS	58
FIGURA 31 – TESTES ENVOLVENDO O FATOR DE ATRAÇÃO	61
FIGURA 32 – TESTES ENVOLVENDO O FATOR DE REPULSÃO	62
FIGURA 32 – TESTE ENVOLVENDO A VORTICIDADE.....	63
FIGURA 33 – NIOS E HARDWARE DEDICADO DENTRO DE UMA FPGA APEX.....	65
FIGURA 34 – TRECHO DO CÓDIGO EM LINGUAGEM EM C.....	66
FIGURA 35 – TELAS COM OS PULSOS DE CLOCKS DAS VERSÕES COM OTIMIZAÇÃO EXECUTADAS NO NIOS $\cong$ 6.602.596 CLOCKS	70
FIGURA 36 – TELAS COM OS PULSOS DE CLOCKS DAS VERSÕES COM OTIMIZAÇÃO EXECUTADAS NO NIOS $\cong$ 2685 CLOCKS	71
FIGURA 37 - RELAÇÃO ENTRE O POLINÔMIO UTILIZADO E A FUNÇÃO SENO	72
FIGURA 38 - RELAÇÃO ENTRE O POLINÔMIO UTILIZADO E A FUNÇÃO COSSENO	72
A) TRAJETÓRIA GERADA USANDO OS POLINÔMIOS	73
B) TRAJETÓRIA GERADA USANDO AS FUNÇÕES SENO E COSSENO.....	73
FIGURA 39 – COMPARAÇÃO ENTRE O USO DAS FUNÇÕES TRIGONOMÉTRICAS E O USO DE POLINÔMIOS.....	73

FIGURA 40 – TRAJETÓRIA, EM MATLAB, DO EXEMPLO IMPLEMENTADO	74
FIGURA 41 – BLOCOS DO TRABALHO FINAL – NIOS + FPU + CONTADOR DE PULSOS.	75

Índice de Tabelas

HISTOGRAMIC IN MOTION MAPPING	IX
TABELA 1 - COMPARATIVO ENTRE AS PRINCIPAIS DIFERENÇAS ENTRE OS MÉTODOS GEOMÉTRICO E O TOPOLÓGICO ..	29
TABELA 2 – COMPARAÇÃO ENTRE OS DIFERENTES MÉTODOS. O TEMPO É MEDIDO EM SEGUNDOS E O CONSUMO DE MEMÓRIA EM OG, UMA MEDIDA RELATIVA CRIADA PELO AUTOR.....	35
TABELA 3 – MÓDULOS E SUAS ESPECIFICAÇÕES	53
TABELA 4 – DESCRIÇÃO DAS VARIÁVEIS USADAS NO MÓDULO <i>ROBO2.M</i>	58
TABELA 5 – DESCRIÇÃO DOS VALORES UTILIZADOS NOS TESTES.	59
TABELA 6 – DESCRIÇÃO DOS VALORES UTILIZADOS NA IMPLEMENTAÇÃO.....	74
TABELA 7 – RESULTADOS OBTIDOS NAS DIFERENTES IMPLEMENTAÇÕES.....	76

Índice de Siglas

ACS	Adaptive Computing System
AGV	Automatic Guide Vehicle
ARMOSH	Aprendizado em Robôs Móveis Via Software e Hardware
ASMR	Aplication Specific Mobile Robot
CLB	Circuit Logic Blocks
CORDIC	COordinate Rotation DIgital Computer
EDA	Electronic Design Automation
ESB	Electricity Supply Board
FAPESP	Fundação de Amparo a Pesquisa do Estado de São Paulo
FPGA	Field Programable Gate Array
GPF	Generalized Potential Fields
GRACO	Grupo de Automação e Controle
HDL	Hardware Description Language
HIMM	Histogrammic In Motion Mapping
LABIC	Laboratório de Inteligência Computacional
OAS	Obstacle Avoidance System
PIO	Pararell Input Output
SOC	System On Chip
SOPC	System On Programable Chip
SRAM	Static Random Acess Memory
TBF	Triangulation Based Fusion
UART	Universal Asynchronous Receiver Transmitter
UNB	Universidade de Brasília
UNB	Universidade de Brasília
VFF	Vector Field Force
VFH	Vector Field Histogram
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits

RESUMO

O crescimento da disponibilidade de ferramentas EDA de interface mais amigáveis, no mercado; O surgimento de linguagens modernas para tal, desenvolvimento (HDL, VHDL e Verilog); A capacidade de validar um hardware antes que se torne puro silício; o aumento do número de portas lógicas em um dispositivo FPGA. Todos esses fatores são contribuintes para que as FPGA se tornem de uso crescente e irreversível. Aliado, a isto, o hardware reconfigurável tem se mostrado uma saída para o desenvolvimento de plataformas específicas. Esta tendência é crescente não somente onde anteriormente já se possuía hardware (placas de rede e roteadores) mas também para gerar soluções específicas para software que eram executados em processadores de uso geral.

Em um outro segmento, os algoritmos de navegação de robôs baseados em células ainda são de alto custo computacional. Dentre os cinco existentes, algoritmos baseados em células, mais conhecidos, o HIMM se provou o melhor. Depois de uma análise verificou-se que HIMM utiliza-se do algoritmo de Campos Potenciais como parte de seu sucesso. Após uma pesquisa bibliográfica e dada à disponibilidade de códigos (cedidos) chegou-se a implementação do algoritmo de Campos Potenciais (parte principal do método HIMM), usado para desvio de obstáculos e planejamento de caminhos locais, em uma FPGA Altera.

ABSTRACT

The growup of offer the EDA's Tools (Electronic Design Automation) with more friendly interfaces in the Marketplace. The criation of news language of development of hardware (HDL,VHDL e Verilog). The capable of test and valid hardware before it make silicon. All of this topics are beneficiary for FPGA and reconfigurable hardware using and growing. Thus, reconfigurable hardware have showed one tendecy for development embeded systems.

By the way, algorithms of navigation for mobile robots, principally, occupancy grids have high computacional cost. In one benchmark realized HIMM (one method for navigation based in grids) show it the better between the four others.

By some search has found that Potential fields was a kernel of method HIMM. Thus this work will put the algorithm Potential Fields in reconfigurable hardware using FPGA, by Altera.

Capítulo 1 – Introdução

1.1 Motivação

Talvez este trabalho isoladamente não represente muito. Entretanto ele faz parte de um outro, muito maior. O ARMOSH [Romero,2000], um projeto em parceria com o LABIC [Labic,2002] financiado pela FAPESP que propõe o estudo e implementação de algoritmos de aprendizado de robôs móveis tanto em software quanto em hardware, visando uma comparação de desempenho em ambas as implementações.

Tendo em mente este universo (algoritmos de robôs móveis + hardware reconfigurável) este trabalho inicia-se pesquisando os algoritmos existentes e procurando um que sirva a tal fim.

1.2 Justificativa

O crescimento da disponibilidade de ferramentas EDA, de interfaces mais amigáveis, no mercado. O surgimento de linguagens modernas para tal, desenvolvimento (HDL, VHDL e Verilog). A capacidade de validar um hardware antes que se torne puro silício. O aumento do número de portas lógicas em um dispositivo FPGA. Todos esses fatores são contribuintes para que as FPGA se tornem de uso crescente e irreversível [Bowen,1999].

O hardware reconfigurável tem se mostrado uma saída para o desenvolvimento de plataformas específicas. Esta tendência é crescente, não somente onde anteriormente já se possuía hardware, (por exemplo, placas de rede e roteadores) [Braun,2001] mas, também para gerar soluções específicas para softwares que eram executados em processadores de uso geral [Oliveira,1999].

Por um outro lado, os algoritmos de navegação baseados em células [Moravec,1996] ainda são de alto custo computacional [Wijk,2001]. Dentre os cinco, algoritmos de navegação

baseados em células, existentes, mais conhecidos, o HIMM [Borenstein,1991a] provou o melhor [Wijk,2001]. Depois de uma análise (Capítulo 5) verificou-se que HIMM utiliza-se do algoritmo de Campos Potencias [Khatib,1986] como parte de seu sucesso. Após uma pesquisa bibliográfica e dada à disponibilidade de códigos (cedidos) chegou-se à implementação do algoritmo de Campos Potenciais em uma FPGA Altera.

1.3 Objetivo

Converter um algoritmo, genérico, de Campos Potenciais (*Potential Fields*), largamente usado em navegação de robôs autônomos móveis, em hardware reconfigurável usando um dispositivo FPGA, Altera.

1.4 Delimitação

Deve ficar claro aqui que o objetivo não é criar um robô físico ou lógico nem mesmo um algoritmo novo. Questionar a potencialidade e viabilidade matemática ou computacional de algoritmos existentes, também não é o interesse.

Assim, o que se pretende é realizar uma escolha dentre os algoritmos existentes através de sua aceitação e abrangência. Uma vez selecionado o algoritmo, implementá-lo em hardware reconfigurável, para ser usado em conjunto com outros trabalhos, dentro de uma (uma mais) FPGA(s) a qual seria o cérebro do projeto ARMOSH. Ao término do trabalho serão realizadas comparações de velocidade x desempenho. Entretanto, essa não é a finalidade principal da proposta. Não existe, nesse primeiro momento, por parte do pesquisador e seu orientador a intenção de criar um processador dedicado mais poderoso que os existentes, hoje em mercado, somente para este algoritmo. É possível, e existe esta meta, que ao término do projeto (ARMOSH), como um todo, e através das vantagens de reconfigurabilidade e possibilidade de paralelismo das FPGAs se consiga vantagens sobre os processadores de uso geral.

1.5 Apresentação

Os passos decorridos da motivação inicial até o objetivo final e sua execução podem ser observados na Figura 1, bem como na descrição a seguir:

Figura 1 – Árvores de escolhas deste trabalho

Capítulo 2: É apresentada uma visão geral da hierarquia e das famílias de robôs existentes hoje no campo comercial e científico, no que se refere à parte física. Uma vez escolhido o robô, físico, um robô autônomo (não-guiado), terreno, com o uso de rodas ou esteiras, a pesquisa a seguir é sob que prisma lógico ele será estudado.

Capítulo 3: Será realizada uma explanação sobre Hardware reconfigurável e FPGA.

Capítulo 4: Apresentação dos três principais paradigmas existentes, no campo lógico da navegação de robôs móveis, onde o que mais se adequou a este projeto foi o paradigma baseado em células.

Capítulo 5: São apresentados os cinco principais trabalhos existentes, sobre algoritmos de navegação baseados em células, fazendo uma comparação de *desempenho x velocidade, prós x contras* e comportamentos em determinadas situações e ambientes. O método escolhido, HIMM [Borenstein,1991], teve suas particularidades estudadas e conclui-se que era baseado no em um método, ainda, mais utilizado Campos Potenciais (*Potential Fields*) [Khatib,1986].

Capítulo 6: É realizada uma apresentação teórica do método de Campos Potenciais, e uma amostra prática através de um código gentilmente cedido por Sérgio Gonsalves Tourino da UNB [Tourino,1999] o qual foi estudado e documentado.

Capítulo 7: São discutidas todas as dificuldades encontradas na implementação do algoritmo de Campos Potenciais em hardware e feitas as comparações entre as várias implementações possíveis.

Capítulo 8: Algumas possibilidades de trabalhos futuros e a conclusão deste trabalho

Capítulo 2 – Robôs Móveis

Um grande sonho da raça humana sempre foi dar vida a suas criações. Após a revolução industrial, passou-se a alimentar a ambição de uma criatura mecânica, inicialmente conhecida como *autômato*, a quem o homem entregaria as tarefas mais pobres e desgastantes. Este conceito foi utilizado pelo escritor Tcheco Karel Capek na peça “Rossum’s Universal Robots”(“O Robô Universal de Rossum”), de 1921. Nesta ocasião ele criou o termo *robot*.

Mais à frente, em 1940, o escritor de ficção científica Issac Asimov concebeu os robôs como autômatos, de aparência humana, entretanto, sem sentimentos. O termo *robótica* foi introduzido por Asimov para representar a ciência devotada ao estudo dos robôs, que seria baseada em três leis fundamentais:

1. Um robô não pode agredir um ser humano ou, por omissão, permitir que um ser humano se machuque.
2. Um robô deve obedecer às ordens dadas por seres humanos, exceto quando tais ordens entrem em conflito com a primeira ordem.
3. Um robô deve proteger sua própria existência, tanto quanto esta proteção não entre em conflito com as duas primeiras ordens.

Nos últimos vinte anos o interesse por robôs móveis, nem sempre de feição humana ou dotado de pernas, tem se tornado objeto cada vez mais crescente, isto se deve ao fato do grande leque de desafios para os quais eles podem gerar soluções.

O universo de robôs pode ser dividido em vários domínios. Um deles se refere ao que de certa forma seria o corpo e o ambiente em que será usado. Desse ponto de vista pode-se classificar os robôs como ilustra a Figura 2.

A primeira distinção importante está entre ser ou não, um robô, guiado. Um robô guiado fica confinado a rádio-controle ou a caminhos predeterminados dentro de uma área de trabalho. Esses caminhos predeterminados podem ser obtidos através de luzes, campos magnéticos, fitas no solo ou qualquer outro mecanismo que induza o robô a uma seqüência lógica de movimentos. O veículo, em nenhuma circunstância, deve deixar este caminho, predefinido. Os AGVs (*Automated Guided Vehicles*) veículos autoguiados - formam esta classe, enquanto que os veículos não guiados não devem estar restrito a nenhum tipo de caminho predefinido.

Figura 2 – Classificação dos Robôs quanto à parte física

O risco e o custo de se manter grandes estoques têm sido estimulados, no que se refere à área de produtos manufaturados, devido às necessidades de pronta-entrega. AGVs podem ser a solução para problemas de transporte dentro destes ambientes, onde partes do produto final devem ser rapidamente transportados entre células de manufatura. [Lazea,1997].

Um segundo campo de aplicação, dentro do domínio AGV, onde a necessidade tem crescido, é o de veículos planetários [Krotkov, 1995]. Nesse caso o robô deverá ser dotado de algum dispositivo que possibilite uma modelagem fiel do ambiente explorado, Figura 3, na maioria dos casos, como esse, são teleguiados. Normalmente, algum tipo de sensor (laser ou sonar) ou processo odométrico (vide 3.2.1), auxilia o controlador nas operações.

Figura 3 – Exemplo de veículo planetário

Robôs para vigília e segurança, como sugerem os filmes de ficção, não têm sido grandes alvos de estudo, pelo menos não no mundo acadêmico. Entretanto, robôs-enfermeiros que acompanham pacientes, na maioria idosos, têm sido bastante incentivados, sobretudo na Carnegie Mellon University [Carnegie,2002].

Como amostra tem-se dois casos de robôs- enfermeiros. No primeiro caso Figura 4 a) [Borenstein,1985] o robô se move conduzido por uma plataforma que tem seus movimentos pré-delimitados, ou seja um AGV. Esse robô foi desenhado para servir pacientes acamados dentro de um ambiente delimitado. Suas funções se limitam a trazer e levar objetos sem que nenhum cuidado médico seja dispensado ao paciente, por ele.

No segundo caso, Figura 4 b) [Montemerlo,2002], *Flo* (nome do robô em homenagem a *Florence Nightingale*, famosa enfermeira da segunda guerra mundial) é autônomo. Dotado de um cinturão de 16 sonares e auxiliado por um laser com uma precisão capaz de medir distâncias de até 5 cm. Além disso, possui conexão externa via rede sem fio, para transmissão de aferições médicas, regularmente.

Figura 4 - a) Robô enfermeiro de Borenstein b) Robô Eldery, codinome *Flo* de um Grupo da CMU

Ainda no caminho de ajuda aos menos favorecidos, robôs que auxiliam deficientes físicos, sobretudo visuais e motores, também têm sido alvo de estudo científico.

A partir de um cinturão de sensores e usando tecnologia experimentada em um robô para desvio de obstáculos - OAS, Borenstein criou um sistema para guiar deficientes visuais, Figura 5, [Borenstein,1990]. O NavBelt, Figura 5 a) é dotado de um vetor de 16 sonares na parte de cima e mais 16 na parte de baixo. O ângulo de abertura, dos sonares, em relação à horizontal do cinturão é diferente, oito dos dezesseis sonares da parte de cima são voltados 30° para cima e os

outros oito apenas 15° , Figura 5 b). Na parte de baixo o mesmo artifício é usado. Tal estratégia permite uma varredura com um ângulo de abertura maior, no sentido vertical, sem, entretanto, desviar atenção de perigos horizontais.

Figura 5 - a) Foto Real do NavBelt b) Figura Explicativa do NavBelt

Cadeiras de rodas para auxílio a deficientes motores também têm sido alvo de estudos, Figura 6, [Borenstein, 1999]. Seguindo a mesma idéia do NavBelt, o NavChair conduz de forma segura pessoas que tiveram os movimentos de pernas e/ou pernas e braços retirados. O deficiente se comunica com a cadeira através de um Joystick, dizendo qual a direção a ser seguida. A cadeira auxiliada pelo cinturão de sonares e o algoritmo VFH, [Borenstein,1998] se conduz através dos obstáculos que tiverem que ser desviados.

Figura 6 - Foto Real do NavChair

Mudando o foco da área de saúde, pode-se encontrar guias e auxiliares de domésticos, com uma eficácia um pouco menor.

Na tentativa de levar a cabo os ideais do escritor Tcheco Karel Capek o robô HERMES[Bischoff ,2000], Figura 6 , um humanóide, foi desenhado para serviços domésticos. HERMES possui dois braços com dois dedos cada braço, Figura 7 b), é autônomo, ou pode ser guiado e utiliza-se de duas câmeras como principais orientadoras para tal. Vários domínios de trabalhos serão possíveis para a plataforma, tais como limpar, transportar, servir, proteger, construção e manutenção de prédios. Entretanto, no momento, HERMES é ainda uma plataforma para que todas essas implementações venham a ser desenvolvidas.

A)

B)

Figura 7 - a) Visão total da lateral do HERMES b) Sequência de movimentos onde HERMES serve uma pessoa.

Um pequeno robô que aspire e limpe toda sua casa enquanto você vai ao shopping pode ser o objeto de desejo de muita gente [Ulrich,1997], Figura 7. Implementado sobre uma plataforma comercial encontrada no mercado (Koala) [K-Team, 2002], Figura 8 a), o ASMR se propõe a realizar a limpeza, mesmo dos cantos mais difíceis.

Figura 8 - a) Foto da base Koala, vendida comercialmente b) Figura Vaccum clear

Pode parecer muito simples a realização de tal tarefa, entretanto, a falta de um protótipo comercial no mercado demonstra que não é assim tão trivial.

Finalizando, este capítulo, não seria possível deixar de lembrar dos robôs guias. Eles vêm sendo usados, na maior parte, em museus. Uma das primeiras experiências nesta área foi RHINO [Thrun,1998], Figura 9 a). RHINO é capaz de realizar uma das tarefas mais difíceis no universo robótica, (vide 3.2.3) locomover-se enquanto constrói mapas, em um ambiente interno e dinâmico (há movimento de pessoas ou objetos). Partindo do mesmo princípio, desenvolvido pela mesma equipe, entretanto acompanhado de uma evolução, pode-se observar MINERVA [Thrun,2000], Figura 9 b). MINERVA possui interface amigável com os visitantes seu “rosto” pode apresentar feições e foi testado em condições reais no “*Smithsonian Museum*” durante duas semanas quando mais de 50.000 pessoas passaram pelo museu. MINERVA estava conectado a Internet de onde o visitante (nessas duas semanas) poderia ver as imagens do museu e até comandar o robô. No que se refere à navegação, MINERVA pode navegar em ambientes estáticos e dinâmicos. Seu sistema de navegação possui uma característica especial que permite navegar em meios a pessoas ainda que sejam ambientes lotados.

A)

B)

Figura 9 – a) RHINO b) Minerva em três ângulos

Depois de todo o exposto, conclui-se que robôs móveis têm sido largamente utilizados nas mais diferentes situações e que a necessidade de pesquisa nesta área é uma constante.

Capítulo 3 – FPGAs e trabalhos relacionados.

Durante os capítulos anteriores, a sigla FPGA foi largamente usada. Vantagens e uso foram descritos, entretanto, o significado real da palavra não foi esclarecido. Os circuitos FPGA são circuitos integrados inicialmente sem funcionalidade, que podem ser configurados pelo projetista e/ou usuário para implementação de uma arquitetura dedicada por ele concebida, Figura 10. Isto é feito pelo carregamento de um arquivo binário (*bit stream*) em um arranjo de células de memória estáticas (SRAM), cujo conteúdo (0 ou 1) define o estado de um elemento reconfigurável do FPGA ou de parte desse elemento. Trata-se de um arranjo matricial de blocos elementares reconfiguráveis (CLBs) circundado por uma densa rede de interconexões. Esta estrutura é, por sua vez, circundada por blocos reconfiguráveis de entrada/saída, associados aos pinos do circuito [Barros, 1996].

Figura 10 - Estrutura Genérica de um dispositivo FPGA

3.1 A computação Reconfigurável

Com a evolução da microeletrônica nos últimos anos, é possível se deparar com uma diversidade de dispositivos eletrônicos mais velozes, com maior capacidade para armazenamento de informações, menor consumo de energia e custos cada vez mais baixos. Dessa forma, os projetistas de computadores ficam frente a frente a um desafio: o balanço entre velocidade de operação dos sistemas versus generalidade de operação desses sistemas [Villasenor,1997].

Pode-se projetar *chips* que executem grande quantidade de funções diferentes, que podem ser caracterizados como de uso geral; entretanto, a velocidade desses dispositivos pode ser lenta, comparado aos dispositivos com aplicações específicas que executam apenas um conjunto limitado de instruções.

Os Microprocessadores como INTEL Pentium ou Motorola PowerPC, comumente encontrados nos computadores pessoais, são exemplos de sistemas com características de uso geral, podendo executar um grande número de funções, atendendo assim um maior número de usuários que fazem uso de computadores pessoais para suas atividades.

Contrário a esses sistemas, pode-se destacar os circuitos com hardware personalizados, conhecidos como ASICs (*Applications-Specific Integrated Circuits*) ou circuitos integrados com aplicações específicas, desenvolvidas para executarem tarefas específicas. Em função dessas especificidades, os projetistas podem desenvolver *chips* menores, mais velozes, consumindo menos energia que um processador de uso geral.

Hoje em dia, na maioria dos hardwares, as funções lógicas são fixas e não podem ser modificadas. No entanto, os FPGAs, podem ser alterados, tanto nas funções lógicas como nos blocos lógicos, além das conexões entre esses blocos, simplesmente enviando sinais de configuração para os *chips*. Os blocos lógicos possuem estruturas similares aos *gate arrays* usados em alguns circuitos ASIC[Wolf,2001], mas são configurados durante a sua fabricação. Ao contrário, os blocos lógicos em um FPGA podem ser re-escritos e re-programados repetidamente, muito depois do *chip* ter sido produzido em uma fábrica.

Recentemente, tem-se desenvolvido novos FPGAs que podem ser configurados com velocidades extremamente altas. Os primeiros FPGAs necessitavam de vários segundos, para alterar seus conteúdos, o que é razoável para verificar novos protótipos em um ciclo de desenvolvimento. Nos dias de hoje, já é possível “hardwares” se adaptarem quase que continuamente em respostas a mudanças de sinais de entrada ou em respostas a sinais do meio externo.

Sistemas computadorizados podem fazer uso dos elementos configuráveis de várias maneiras. Uma das mais comuns, seria comutar o hardware para diferentes funções a partir de comandos que seriam semelhantes a fechar um programa e reiniciar outro. O tempo de reconfiguração do hardware seria aceitável para determinadas aplicações. Isso poderia ocorrer também de forma dinâmica, de tal maneira que a reconfiguração ocorresse entre tarefas sendo executadas por aquele FPGA.

Computação Reconfigurável, ainda é uma área em desenvolvimento, diferente de arquiteturas de computadores onde os conceitos e mecanismos vêm sendo longamente testados e comprovados. Portanto, definir e validar esses tópicos em computação reconfigurável, se torna um desafio para os pesquisadores atualmente [Dehon,1996].

Em termos básicos, a computação reconfigurável combina a velocidade do hardware com a flexibilidade do software. A tecnologia da computação reconfigurável consiste na habilidade de se modificar o hardware da arquitetura para esse se adequar à aplicação [Villasenor ,1997] [Star,2001]. Essa reconfiguração do hardware pode ser realizada de duas maneiras:

- Reconfiguração Estática: onde a utilização do *chip* deve ser suspensa, até que a o processo de reprogramação seja concluído.
- Reconfiguração Dinâmica: onde uma parte do *chip* pode continuar sendo utilizado durante a reprogramação. São reprogramadas as regiões do chip que não estiverem sendo usadas naquele determinado momento, e as regiões em uso se mantêm inalteradas.

3.2 Reconfiguração Dinâmica

Os dispositivos FPGAs reconfiguráveis dinamicamente permitem que uma parte selecionada da lógica programável e dos recursos de roteamento sejam atualizadas enquanto o restante dos recursos continuam a funcionar [Burns,1997] [Lysaght,1993].

Como os tempos de configuração não são desprezíveis, a habilidade de intercalar execução e reconfiguração, sem prejuízo do desempenho, é uma questão que ainda merece atenção e esforços de pesquisa. Com a introdução de FPGAs com tempos de reconfiguração mais rápidos, e suporte à reconfiguração parcial é possível usar FPGAs em um ambiente de reconfiguração dinâmica. (RTR, *Run-Time ReconFiguration*) .

Figura 11 - Esquema de Reconfiguração

Um sistema de reconfiguração dinâmica inclui pelo menos uma área de reconfiguração, onde blocos lógicos podem ser carregados em tempo de execução. A Figura 11 ilustra a reconfiguração dinâmica de um sistema composto de cinco circuitos ou tarefas. A tarefa A é uma função de entrada que alimenta as tarefas B, C ou D, que alimentam a tarefa E que é a função de saída do sistema. As tarefas de entrada e saída são permanentemente residentes no FPGA enquanto as três tarefas dinâmicas alternam-se sob o controle de um sinal de *Swap*.

A capacidade de reconfiguração dinâmica permite o compartilhamento ao longo de tempo de tarefas diferentes, o que pode reduzir significativamente a área de silício exigida. Esta tecnologia torna possível o conceito de hardware ilimitado ou “Hardware Virtual” [Cardoso, 2000].

A reconfigurabilidade das FPGAs no caso de algoritmo de Campos Potenciais, pode-se ser usada para redefinir o tamanho das células do mapa e as variáveis (Tabela 4) de acordo com o ambiente e os obstáculos.

3.3 Sistemas de Computação Reconfigurável

Os sistemas de computação reconfigurável são plataformas onde a arquitetura pode ser modificada em tempo real para melhor se adequar à aplicação que será executada. Deste modo, o processador reconfigurável passa a trabalhar com uma arquitetura desenvolvida exclusivamente para aquele determinado tipo de aplicação, permitindo uma eficiência muito maior do que a normalmente encontrada em processadores de uso geral. Isto ocorre porque o hardware é otimizado para executar os algoritmos necessários para aquela aplicação específica. Ou seja, para se obter um melhor desempenho de um algoritmo, este deve ser executado num hardware específico para ele. Acentuados ganhos de desempenho podem ser obtidos com a utilização deste hardware específico.

O hardware reconfigurável pode ser classificado em três categorias:

- Hardware Puro (hardware para aplicações específicas)
- Coprocessadores (*chips* específicos para determinadas tarefas)
- Computadores (plataformas de computação completamente reconfiguráveis)

Atualmente, a forma de utilização mais difundida dos *chips* reconfiguráveis é como parte de sistemas que utilizam processadores tradicionais. Os *chips* reconfiguráveis atuam como coprocessadores trabalhando com algoritmos pesados e específicos. Dessa forma aliviando a carga do processador principal, permitindo que este possa executar outras tarefas mais genéricas. Nesses casos, o chip reconfigurável pode ser comparado com um *full-custom* do sistema. Ele se

adapta em tempo real para tarefas específicas e pesadas que certamente exigiriam grande tempo de processamento do microprocessador principal.

Muitos trabalhos têm sido realizados na construção de arquiteturas reconfiguráveis [Razdan,1994] [Wirthlin,1995] [Hauser,1997] [Taylor,1996] , entre vários outros. Atualmente a área de computação reconfigurável apresenta-se como uma tecnologia extremamente interessante tanto no campo industrial, como no campo acadêmico. São inúmeras as atuais aplicações de computação reconfigurável, como por exemplo: telefones celulares, controladores de dispositivos presentes em carros e aviões, hardwares específicos de alto desempenho para as mais diversas aplicações, robótica etc.

3.4 O Kit de Desenvolvimento Excalibur

O Excalibur® é um conjunto de hardware (placa de desenvolvimento, FPGA, periféricos e dispositivos de conexão) e software (ferramentas para desenvolvimento e testes), para o desenvolvimento de hardware. Além do processador Nios *softcore* e, futuramente, dos *hardcores* ARM e MIPS, o Excalibur possui todas as ferramentas que engenheiros de hardware e software precisam para o desenvolvimento de sistemas de alta performance em Dispositivos de Lógica Programável. Este kit é composto de um processador PLD-otimizado que os projetistas podem implementar imediatamente, diminuindo o tempo de desenvolvimento de sistemas de hardware.

A ferramenta de desenvolvimento Excalibur é composta pelos itens:

Processador RISC configurável Nios core e periféricos

Compilador GNUPro® da Cygnus®, uma parceria Red Hat®

O software de desenvolvimento e testes Quartus®

Placa de desenvolvimento com o FPGA APEX EP20K200E

3.5 O Processador Nios

O processador embutido Nios® é um *softcore* de um processador RISC configurável com um conjunto de instruções de 16 bits e um caminho de dados selecionável de 32 ou 16 bits. Tendo sido desenvolvido especificamente para lógica programável, o Nios é otimizado para ocupar uma reduzida área do PLD e pode prover até 50 MIPS de desempenho. Inicialmente sendo destinado para a família de dispositivos APEX da Altera, este processador ocupa apenas 12% da área de um dispositivo de 200 mil portas (por exemplo, EP20K200E). Considerando que a família APEX já possui dispositivos com 1,5 milhão de portas, isto significa que há uma infinidade de recursos disponíveis para o restante do sistema, Figura 13.

Figura 13- Diagrama de Blocos do Processador Nios®

A eficiência de uso do dispositivo possibilita também que múltiplas instâncias do processador possam ser usadas para aplicações de alto desempenho que requerem implementações com múltiplos processadores, como mostra a Figura 12.

Figura 12 - Múltiplos Nios em um APEX

O processador Nios pode ser parametrizado, permitindo que o usuário faça o balanço entre custo e desempenho, sem a necessidade de ser um especialista em projeto de processadores. Apesar da configuração, o conjunto de instruções permanece o mesmo, possibilitando o uso de ferramentas de desenvolvimento de software padrões de mercado como compiladores C/C++.

3.6 Compilador GNUPro

O compilador e *debugger* GNUPro da Cygnus, uma companhia da Red Hat, é uma ferramenta padrão, usada por fabricantes de software de todo o mundo. Ele é uma ferramenta de desenvolvimento C/C++. Otimizado para o processador embutido Nios, o GNUPro proporciona um ambiente de desenvolvimento familiar aos profissionais da área, incluindo:

- Compilador otimizado C/C++
- Assembler GNU
- Debugger interno
- Utilitários binários

3.7 Quartus II ® – Ferramenta de Desenvolvimento

O software de desenvolvimento Quartus II permite aos projetistas, o processamento de milhões de portas lógicas, com vantagens nunca vistas em ferramentas de desenvolvimento de PLDs. O software Quartus suporta soluções em nível de sistema com editoração de blocos,

trabalho em grupo e suporte para megafunções. Além disso, um sistema de análise lógica, embutido, permite aos engenheiros verificar a funcionalidade e a temporização do *chip*, observando sinais e valores internos gerenciados pelo clock interno do sistema.

3.8 Placa de Desenvolvimento

O pacote Excalibur® inclui ainda uma placa de desenvolvimento que proporciona aos engenheiros uma plataforma de hardware através da qual, podem iniciar o desenvolvimento de um sistema sobre o *chip* programável.

A placa de desenvolvimento pode ser expandida para suportar uma vasta gama de interfaces através de conectores de protótipos, conectores PCI ou SDRAM.

O chip FPGA que acompanha o kit de desenvolvimento pertence a família APEX da Altera. A família APEX representa a última palavra em tecnologia reconfigurável, possui um nível de integração de até 0.15 μ e um equivalente a mais de 2.000.000 de portas lógicas, além de mais de 400k bits de memória RAM.

O modelo que atualmente acompanha o Kit Excalibur é o APEX 20k200E com 526.000 portas lógicas e 106.496 bits de memória RAM (interna ao chip).

3.9 Paradigma SOC

O Paradigma SOC (*System on a Chip*), consiste no desenvolvimento de sistemas completos em um único *chip*. Ou seja, em apenas um *chip* são embutidos o(s) processador(es), memória, temporizador(es), interfaces etc. Dessa forma, todos os circuitos necessários para o funcionamento de um determinado sistema são implementados em um único dispositivo. Essa idéia tem sido amplamente adotada pela maioria de fabricantes da área de hardware. Devido ao seu excelente desempenho e versatilidade, a tecnologia FPGA tem se mostrado perfeita para implementar os sistemas SOC.

A ferramenta de desenvolvimento Excalibur inclui referência de projetos que são uma completa configuração SOC, incluindo um processador-core, alguns periféricos e um barramento *on-chip*. Estes projetos de referência são pré-carregados em uma memória FLASH *on-board*,

permitindo aos projetistas iniciar o desenvolvimento de seus códigos imediatamente. Os periféricos incluídos no projeto de referência são:

- UART
- Timer
- Interface para memória externa (SRAM, FLASH)
- Interface para memória interna (ESB)
- PIO (*Parallel I/O*)

O processador Nios® comunica-se com os dispositivos periféricos através do Módulo de Barramento Periférico. Este componente é gerado automaticamente pelo software *SOC builder*, baseado nos periféricos requisitados pelo usuário. Isto inclui:

- Codificação de endereço para cada periférico
- Conexão do barramento de dados entre cada periférico e o Nios-core
- Controle de interrupções para todos os periféricos

Um módulo de barramento de tamanho variável, permitindo interfaces diretas para periféricos de 8-, 16-, e 32-bit. A Figura 14 apresenta um exemplo de um sistema projetado com para trabalhar como o Sistema Excalibur da Altera [Altera,2002].

Figura 14 - Referência de Projetos de Periféricos SOPC

4.0 Trabalhos relacionados

Outros pesquisadores também já perceberam que o uso das FPGAs em algoritmos mais complexos, tem sido causa de sucesso. Sobretudo, para a área específica de robótica, os resultados obtidos têm sido satisfatórios, para não se dizer brilhantes [Razdan,1994] [Wirthlin,1995] [Hauser,1997] [Taylor,1996] [Wolf,2002] [Giorgini,2002] [Teixeira,2002] [Ribeiro,2002]. Em 1998 um pesquisador já criava um compilador capaz de fragmentar códigos fontes que poderiam ser processados em paralelo para o uso em FPGAs e processadores RISC [Gokhale,1998], com a finalidade de trabalhar algoritmos de robótica. Além dessa área, processamento de sinais, tratamento de imagens, controle e manipulação de bits, compactação e encriptação de dados [Saxena, 2000] também têm ido alguns dos alvos.

Em Shu-Yi Yu, Nirmal Saxena e Edward J. McCluskey [Yu,2000] os autores propuseram um trabalho sobre um dispositivo FPGA chamado ACS da empresa Xilinx. Nesse caso ao invés de usar FPGAs para reduzir o tempo de resposta de algoritmos críticos, os dispositivos FPGAs são utilizados para aumentar a tolerância a falhas em um robô. Aumentar a tolerância a falhas requer redundância de hardware. Isso obviamente requer mais tempo de resposta. Em um hardware seqüencial sim, entretanto, os pesquisadores aqui usam FPGAs trabalhando em paralelo. Assim, múltiplas cópias do mesmo algoritmo são executadas em paralelo e depois seus resultados comparados. O trabalho pretende com isso aumentar a independência do robô para que o mesmo possa vir a trabalhar em lugares onde a presença do homem para a correção de erros nem sempre é possível.

Não é difícil perceber que a obtenção de velocidade é apenas uma das vantagens que as FPGAs podem trazer. Em Angkul Kongmunvattana e Prabhas Chongstitvatana [Kongmunvattana,1998] é visível algumas vantagens do uso de FPGAs:

- 1) Ser projetadas para tarefas em paralelo, o que lhes garante uma melhor capacidade para executar tarefas em tempo real, visto que isto nem sempre é obtido por um único processador de propósito geral.

2) Baixo consumo de energia.

3) Capaz de trabalhar dentro da arquitetura não convencionais (non-Von Neumann)

Capítulo 4 – Navegação de robôs móveis

4.1 Domínio físico

Fazendo uma análise sobre os exemplos que este trabalhando vem demonstrando auxiliado pela visão geral que a Figura 2 proporciona, pode-se concluir que uma área crescente de pesquisa vem sendo: Robôs Móveis – Não Guiados (Autônomos) – Terrenos – Com Esteiras ou Rodas.

Os modelos aquáticos [ROV,2002] e aéreos [Laine,1998] [Saripalli,2002] são bem mais raros. Os modelos com pernas são assunto comercial de bastante interesse para os Japoneses [Fujitsu,2002] [Honda,2002] [Kawada,2002]. Dessa forma, os estudos realizados nessa área, robôs com pernas, tornaram-se segredo industrial (o que é fácil perceber dada a falta de artigos a respeito) o que dificulta por demais a pesquisa nesse campo.

Como a intenção desse trabalho, definida previamente (vide Capítulo 1), não é a de explorar caminhos poucos trilhados ou mesmo criar algo extremamente inusitado, mas sim seguir uma linha já existente. Assim, no que se refere ao primeiro corpo do robô, domínio físico, esta pesquisa será desenvolvida com Robôs Autônomos, terrenos, com o uso de rodas ou esteiras.

Agora, já definido o meio físico, para onde este trabalho será conduzido, resta definir o meio lógico.

4.2 Domínio Lógico

Uma vez particionado o domínio físico, o lógico pode, dependendo do prisma, ser dividido em vários subdomínios. Pode-se ainda dentro do domínio físico escolhido, escolher subdivisões dentro da área de inteligência artificial, visão computacional ou processamento de alto desempenho e distribuído. Entretanto, como o o domínio físico escolhido é um robô *móvel e*

autônomo a melhor escolha para divisão em subdomínios lógicos é no que se refere aos tipos de tarefas e dificuldades dentro do campo da navegação.

4.2.1 Posicionamento odométrico

Neste tipo de tarefa o robô tem conhecimento do ambiente através de um mapa (*Occupancy Grids* ou Mapa Topológico, vide mais adiante) previamente fornecido. Também é do seu conhecimento a posição inicial em que ele se encontra. O problema aqui, para o robô, é manter-se localizado dentro do mapa durante a sua locomoção. Os decodificadores colocados nas rodas são decrementados ou incrementados a medida que o robô caminha e devem ser validados por meio do apoio de outras medidas uma vez que, como visto anteriormente, existe bastante imprecisão nos sensores. Uma ferramenta largamente usada neste domínio é o filtro de Kalman [Kalman,1960].

4.2.2 Localização Global Dentro de um Mapa

Aqui o robô também possui um mapa como aquele anteriormente referenciado. Entretanto, ao invés de completo ele tem pontos de referências. Complicando ainda mais, dessa vez ele não sabe onde está em relação ao mapa e sua tarefa é exatamente essa. Uma Solução usando uma mistura de câmeras, lasers, sonares com rotação [Schiele,1994] foi usada em 1994, e apesar de todo o aparato não apresentou resultados satisfatórios. Uma solução robusta e precisa para grandes ambientes ainda não foi encontrada [Wijk,2001].

4.2.3 Construção de Mapas e localização simultânea

A maioria dos pesquisadores considera esse problema como sendo o mais complexo. O robô, a princípio, não tem nenhuma informação do ambiente. O robô aqui deve explorar e construir seu próprio mapa enquanto se situa dentro dele. Isto leva ao dilema de o que fazer primeiro? Para uma localização acurada o robô precisa de um mapa. Entretanto, ele mesmo tem que construir tal mapa.

Assim, o robô, ao caminhar, tem que construir seu próprio mapa, atualizar e verificar em tempo integral. É necessário ter a certeza de que esta construção vem a ser a mais perfeita possível. Esta atualização, enquanto desvia e caminha, torna esta tarefa de altíssimo custo computacional [Wijk,2001].

4.3 Os Paradigmas

A fim de resolver os domínios em questão existem dois grandes e clássicos paradigmas; o *geométrico* e o *topológico*.

Nos dois primeiros domínios é muito comum o uso de paradigmas baseados em células. No último, além desse paradigma é também não raro o uso de paradigmas híbridos.

4.3.1 Paradigma baseado em Células – Geométrico

Em robôs móveis, um mapa de células é uma matriz bidimensional do ambiente onde o robô se encontra e a posição do robô é mantida através de coordenadas (x,y). À medida que o robô vai andando de um ponto a outro no ambiente, sensores captam o ambiente e atualizam a matriz. Assim, as células vão sendo preenchidas como ocupadas, livres ou não exploradas. Essa matriz de células vai sendo construída e atualizada. Na maioria dos casos o mapa de células é gerado pelo robô enquanto explora o ambiente sendo atualizado recursivamente [Wijk,2001].

O conceito de mapas baseados em células para robôs móveis começa a surgir em 1983, na Carnegie Mellon University [Moravec,1996]. O método de atualização e preenchimento das células pode variar sendo: probabilístico (bayesiano) o mais usado [Moravec,1985], Dempster-Schafer [Elfes, 1989], “Fuzzy Set Theory” [Yamauchi,1998], TBF (Triangulation Based Fusion) [Wijk,1998], HIMM (Histogramic In-Motion Mapping) [Borenstein,1991] os mais comuns. Uma comparação, dentre os cinco, foi feita recentemente por Wijk [Wijk,2001].

Em algumas abordagens o paradigma pode ser usado com mapas já previamente construídos, e/ou “*landmarks*” apenas para localização [Ulrich, 1991]. Em outras, as células podem conter mais de um tipo de informação, Konolige [Yamauchi,1998].

Como todas as soluções este paradigma apresenta seus prós e contras, vide Tabela 1. Quando ocorre a atualização de células com a informação vinda dos sensores, uma grande quantidade de cálculos deve ser realizada. Embora, algumas criações de dependências e correlações possam tornar isso mais rápido e leve [Yamauchi,1998], ainda assim, o uso de matriz de células, envolvem um alto custo computacional.

Tabela 1 - Comparativo entre as principais diferenças entre os métodos geométrico e o topológico

Geométrico	Topológico
O reconhecimento de lugares parecidos não é problema uma vez que sua posição é mantida em relação ao ponto de partida.	Reconhecimento de lugares parecidos torna-se difícil, sobretudo se não existe o uso de “ <i>landmarks</i> ” e/ou outra técnica de apoio.
Requerem determinação precisa da posição do robô, que devido à falta de odometria, citada anteriormente, constitui uma tarefa complicada.	Não requer determinação acurada da posição do robô.
É fácil de se construir mas não de manter.	É difícil de se construir e manter.
Dependendo do tamanho do ambiente, da configuração de células e da capacidade de memória, planejar pode se tornar um problema grave.	Permite eficiente planejamento com pouca complexidade e baixo custo computacional.

4.3.2 Paradigma topológico

Este paradigma [Kuipers,1991] [Engelson,1992] [Mataric,1992] [Borghi,1995] usa um grafo de adjacência como mapa de localização do robô. O robô é situado no mapa como sendo um dos nós do grafo. Nesta abordagem, um dos maiores problemas, é se situar localmente, ou seja, o robô sabe onde está no contexto geral, mas pode não saber em qual ambiente se encontra, sobretudo, se houverem salas parecidas. Há aqui, um custo computacional mais baixo do que na abordagem anterior. Entretanto, essa abordagem sozinha não pode garantir uma interpretação

geométrica detalhada do ambiente, bem como da posição do robô dentro deste ambiente [Kortenkamp,1994]. Como o paradigma anterior, também tem suas vantagens e desvantagens, Tabela 1.

4.3.3 Paradigmas Híbridos

Não é difícil notar que os paradigmas geométrico e topológico se complementam. As fraquezas de um, são os pontos fortes do outro, e vice-versa [Thrun,1996]. Assim, como também, não é difícil encontrar literatura de vários pesquisadores que fundiram os dois paradigmas, cada qual a sua maneira. Dentre eles podem ser encontrados os trabalhos a seguir:

Em Thrun & Arno [Thrun,1996] existe a proposta de uma construção de mapas baseados em células e mais tarde mapas topológicos construídos sobre estes, para a construção de um mapa global. Os mapas baseados em células são construídos com o auxílio de redes neurais, usando o método “Back Propagation” para mapear os sonares e assim codificar as células. A exploração e aprendizado são feitos através do custo mínimo do caminho ainda não explorado. Uma vez que o mapa de células esteja montado, ocorrerá a fusão.

Os espaços livres dentro dos mapas baseados em células, são particionados em pequenas regiões, separados por linhas críticas. Essas linhas correspondem às passagens mais complexas tais como: portas e corredores mais estreitos. Então, a região particionada é mapeada dentro de um grafo isomórfico. Partindo de um mapa baseado em células e chegando a um grafo topológico cinco passos são feitos, como pode ser visto na Figura 10.

a) Mapa Métrico b) Diagrama de Voronoi c) Pontos Críticos d) Linhas Críticas e) Regiões topológicas f) Grafo topológico

Figura 15 - Extração de mapas topológicos sobre mapas baseado em grids

Por sua vez, Eric Dedieu e Jose Del R. Millán [Dedieu,1998] partindo do mesmo princípio de Thrun (mapas construídos de forma híbrida e usando redes neurais para a construção do mapa baseado em células), propõe uma abordagem diferente. Segundo os autores, eles não estão interessados em construir um mapa global acurado do ambiente, o que requer grande esforço computacional, sobretudo se o ambiente é dinâmico. A proposta é que os mapas, topológico e baseado em células, trabalhem em paralelo, Figura 16, e não hierarquicamente ou sobrepostos, como propunha o trabalho anterior.

Os mapas topológicos serviriam para a conexão de cada ambiente mapeados em células anteriormente, dando assim uma maior flexibilidade de planejamento global. A métrica imediata ou reativa seria obtida através de mapas de células de ocupação, as quais iniciariam com uma resolução finíssima que, mais tarde, seria adequado à resolução do ambiente, tão logo se souber o tamanho do mesmo.

Figura 16 - Mapa baseado em Grid construído em paralelo ao topológico.

A literatura é vasta em exemplos nesta área. Tom Duckett e Alessandro Saffiotti [Duckett,2000] defendem que mapas baseados em células, para localização global, em ambientes grandes e divididos em várias salas, requer muito esforço computacional. A saída, assim, seria trabalhar com mapas topológicos. Entretanto, estes não provêem um detalhamento geométrico para o reconhecimento do local onde o robô se encontra.

Assim, a proposta é uma integração das duas técnicas, para obter uma localização global precisa, sem recorrer a sensores altamente acurados e um grande esforço computacional. O caminho para tal integração seria, resumidamente: no nível mais baixo, construir matrizes de células para cada ambiente em particular. No nível intermediário, esses mapas seriam conectados para montar um mapa topológico. Em um último nível, depois de passar pela “Relaxation Technique” [Duckett,1999] e tornar a informação consistente o bastante, esse mapa topológico seria convertido em um mapa de células global, Figura 16.

Enfim não nos faltam exemplos sobre paradigmas híbridos e suas diferentes abordagens. Existem, ainda, abordagens não híbridas, mas que dão um enfoque diferente às abordagens tradicionais citadas, principalmente o geométrico [Youngblood, 1999] [Schultz,1998] [Schiele,1994]. A relevância para este trabalho é que, tais paradigmas foram, e ainda são, largamente usados em exploração, navegação e planejamento, juntos ou isoladamente. Lembrando ainda que todos eles frisam de uma forma ou de outra que construir mapas é uma

atividade que requer grande esforço computacional, seja qual for o paradigma. Tom Duckett [Duckett,2000] cita que Thrun [Thrun,1998a] leva duas horas para construir um mapa baseado em células com uma resolução de 1 metro por células em um local de 90 x 90 metros. Assim, o objetivo desses pesquisadores quando misturaram cada um dos paradigmas, quase sempre é a diminuição de tempo de resposta. Uma vez que tais robôs, na sua maioria, trabalham em ambientes de tempo real.

Figura 17 – Extração de um mapa topológico da mapa de células.

4.4 Resumo

Finalizando este capítulo conclui-se pelas escolhas de um algoritmo para robô autônomo, terreno, com o uso de roda ou esteira (4.1) para localização global dentro de um mapa (4.2.2) usando portanto o paradigma baseado em células (4.3.1) o qual se demonstrou melhor para essa situação. Um outro trabalho dentro do projeto ARMOSH [Scatena,2002] será realizado dentro do contexto paradigmas topológicos a fim de que o robô pronto, mais tarde, possa ter as duas opções. No capítulo 5 é realizada uma comparação de velocidade e precisão para as diferentes tarefas (corredor e sala) dos principais algoritmos existentes baseados em células.

Capítulo 5 – Os Tipo de Mapas de Células.

Na tentativa de avaliar os métodos baseados em mapa de células existentes no mercado, seus prós e contras, depara-se com um trabalho bastante esclarecedor, Olle Wijk [Wijk,2001]. Neste trabalho um método novo é apresentado pelo autor, o TBF.

Seu método é comparado a outros quatro já existentes sob três aspectos: desempenho, velocidade e custo computacional. Três dos métodos são baseados na teoria Bayesiana: Bayesiano, Dempster Shafer e Fuzzy. O quarto método escolhido foi o HIMM [Borentein,1991a], baseado na teoria dos Campos Potenciais.

Dois experimentos diferentes foram feitos usando cada um dos cinco métodos. Um ao longo de um corredor e outro em uma sala com mobília, vide Figura 18. Em todos os casos o robô utilizado foi o mesmo um Nomad 200 com 16 sonares. Na sala foram realizadas 733 (leituras) x 16 (número de sonares)=11728. No corredor 915 (leituras) x 16 (número de sonares)=14640. Esses dados foram armazenados.

Figura 18 – a) Vista superior da sala b) Vista superior do corredor. Em ambos os casos a linha representa a trajetória do robô.

Assim as comparações realizadas foram feitas em um PC (*Path Planning off-line*) e não no Nomad 2000. Os resultados obtidos nas comparações podem ser observados na Tabela 2.

Tabela 2 – Comparação entre os diferentes métodos. O tempo é medido em segundos e o consumo de memória em OG, uma medida relativa criada pelo autor.

Processador 450 Mhz - PC			
Método	Corredor	Sala	Consumo de Memória
TBF	9 s	9 s	1 OG
Bayes	17 s	11 s	1 OG
Dempster-Shafer	25 s	17 s	2 OG
Fuzzy	31 s	24 s	3 OG
Borenstein	0.3 s	0.2 s	1 OG

Partindo da Tabela 2 é possível perceber que o método Fuzzy tem o maior custo computacional acompanhado do pior tempo. O Método HIMM, Borenstein, é o de melhor tempo e custo.

5.1 Desempenho na Sala.

O tamanho da célula utilizada no experimento da sala foi de 0.1m .A Figura 19 a) mostra o rastreamento do sonar utilizado na sala. Cada ponto nessa plotagem representa o ponto central do arco formado pela leitura do sonar. Os vários pontos que aparecem no centro da sala representam uma pessoa que passou se movimentando lentamente enquanto o robô Nomad 200 mapeava o ambiente.

A finalidade de incluir um dado dinâmico no ambiente que estava sendo mapeado, era investigar a capacidade de cada método, em distinguir entre objetos dinâmicos e estáticos. A idéia de que seres humanos são frequentemente encontrados em “ambientes internos”, também, justifica o acréscimo ao experimento.

Se o mapa de células construído tiver o propósito de localização, é desejável que esse mapa possa filtrar os objetos que estejam se movendo. Por um outro lado, se o mapa for construído para planejamento de caminhos locais, que é o caso de Borenstein, o objetivo seria que este método captasse, de forma clara, os dois tipos de objetos, dinâmicos e estáticos. No caso deste experimento isto ocorre, como pode ser percebido na Figura 19 f) no canto superior direito.

No mapa com o método TBF, Wijk, Figura 19 b), duas observações devem ser ressaltadas. Primeiro, várias superfícies na sala, como paredes e a mesa de jantar, são modeladas de forma clara no mapa. Segundo, o rastro feito pela pessoa que se move é eficientemente apagado. Comparado aos outros métodos, o mapa de células TBF é esparsos, mas ainda assim nos leva a uma descrição do ambiente.

No mapa de células Bayesiano, Figura 19 c), as áreas cinzas correspondem a áreas desconhecidas, onde não existem leituras atualizadas no mapa. O mapa Bayesiano, assim como o TBF, também falha na representação onde era exigida maior precisão. A mesa maior (de jantar) não foi sequer modelada e a menor teve somente um de seus cantos representado. Isso acontece porque as cadeiras causam embaralhamento das leituras do sonar. Também aqui, como era desejado, é possível ver que nenhum dado sobre a pessoa que se movia foi captado.

O mapa de Dempster-Shafer, Figura 19 d), é muito similar ao Bayesiano. Esse método também falhou na representação das mesas, assim como o anterior e pelos mesmos motivos. Entretanto, a representação das paredes aqui pode ser vista de forma bem mais efetiva.

O mapa Fuzzy, Figura 19 e), é um dos mais fieis, no que tange a modelagem. Nenhum segmento de linha ficou faltando. Entretanto, ele possui uma certa dificuldade em modelar espaços vazios. Observando-se as passagens de portas pode se notar que este método foi o mais incerto nesse tipo de modelagem. O método deveria livrar-se do rastro deixado pela pessoa, mas, não obteve êxito.

Finalmente, o mapa obtido através do método HIMM, Borenstein, Figura 19 f), demonstra uma surpreendente – já que foi o mais rápido – descrição do ambiente. Todos os objetos foram razoavelmente modelados. Como era esperado – o modelo deve mapear objetos estáticos e dinâmicos – os dados da pessoa que se movimenta são mapeados.

Cabe salientar que os únicos métodos capazes de modelar, com perfeição aceitável, a mesa de canto do sofá foram o Fuzzy e Borenstein.

Figura 19 - a) Dados do sonar usados no experimento b) Mapa TBF c) Bayesiano d) Dempster e) Fuzzy f) Borenstein.

5.2 Desempenho no Corredor

O tamanho da célula usada no experimento do corredor foi de 0.15 m. O rastreamento realizado pelo radar, usado para avaliar os métodos, pode ser visto na Figura 20 a). A linha que aparecer no decorrer do centro do corredor é, novamente, uma pessoa que caminha sempre à frente do robô. Para melhor investigar o comportamento dos métodos avaliados, todas as portas de acesso ao corredor foram deixadas abertas, durante a leitura original.

Figura 20. a) rastreamento realizado pelo sonar no experimento corredor b) TBF c) Bayesiano

A Figura 20 b) mostra a modelagem do corredor sob o prisma do método TBF. Note que todos os detalhes do corredor foram bem delineados. O sinal da pessoa que se movia foi totalmente removido.

O método Bayesiano pode ser observado na Figura 20 c). A partir desta imagem pode-se notar que muitas partes do corredor, sobretudo as próximas às portas, não foram bem captadas. Entretanto as passagens foram bem detectadas e o sinal da pessoa também foi totalmente removido.

Na Figura 21 a) pode-se ver o experimento realizado no corredor sob o método Dempster-Shafer. Este mapa é um pouco melhor que o fornecido pelo método Bayesiano. Entretanto, vários segmentos das paredes ainda estão faltando. Algumas sobras atrás das paredes foram incorporadas ao mapa. Assim como no método anterior as passagens foram bem modeladas e o sinal, deixado pela pessoa, retirado.

Figura 21 - a) Dempster-Shafer. b) Fuzzy. c) Borenstein HIMM.

O mapa obtido pelo método Fuzzy pode ser observado na Figura 21 b). Assim como no primeiro experimento, o espaço livre modelado pelo métodos é menor do que o mundo real. Observe que o corredor torna-se mais estreito do que o apresentado em outros métodos. Além disso, boa parte dos rastros deixados pela pessoa não foram removidos.

Finalmente, o mapa obtido pelo método Borenstein, Figura 21 c). Este método, assim como o Bayesiano e Dempster-Shafer, teve dificuldades em manusear todos os segmentos da parede. Entretanto, a representação de mundo obtida é bastante aceitável, incluindo o rastro deixado pelo teste dinâmico.

5.3 Resumo e discussões

Os cinco métodos apresentados produzem mapas com características particulares. Sabe-se qual o mais rápido, Tabela 2. Entretanto, ainda não está claro qual o melhor, no que se refere a desempenho, tanto em corredor quanto em sala. Para responder esta questão é preciso que se saiba com qual finalidade e sob quais condições o mapa seria construído. Conforme ver-se-á a seguir, cada método deve melhor se adequar a um tipo de situação.

TBF: O Mapa obtido aproxima-se do rastreamento obtido pelo sonar. A qualidade do mapa, em comparação aos outros, fica acima da média. Os pontos obtidos pela interpretação da leitura original são esparsos. Entretanto, isto pode ser melhorado se houver um acréscimo dos pesos no algoritmo original. Estes fatos fazem com que o método TBF seja adequado para localização mas, não seja recomendável para planejamento, sobretudo, em ambientes dinâmicos.

Bayesiano e Dempster-Shafer: As duas abordagens geraram um tipo de mapa com características bastante parecidas. Os melhores resultados são obtidos em lugares onde existe a menor incidência de reflexões do sonar que, quando ocorrem, geram dados em camadas, como se fossem sombras tal situação pode ser melhor observada no experimento do corredor. Infelizmente essas “camadas de sombras” são muito comuns em sonares. Assim, esse método somente poderia tornar-se mais eficaz se fosse adequado para lidar com essa limitação ou se fosse acoplado a outro tipo de sensor.

Fuzzy: Aqui os problemas com “camadas de sombra” são bem menores. Entretanto paga-se o preço por um método bastante rigoroso, que acaba por sacrificar a representação de espaço livre. Uma construção com graus de certeza/incerteza mas baixos com relação a questão ocupado/livre é sugerida pelo autor como melhoria do método.

Borenstein: Aqui pode-se obter uma das mais satisfatórias descrições dos ambientes. Entretanto, também se encontra aqui a dificuldades em lidar com as “camadas de sombra”. O mapa obtido é adequado à representação de objetos estáticos e dinâmicos e possui um baixo custo computacional. Este fatos fazem dele um método ótimo para planejamento local rápido.

5.4 HIMM x Campos Potenciais. Qual a relação?

Foi descrito, detalhadamente, nos itens anteriores (5.1 e 5.2) que o melhor método em vários aspectos é o mapa de células de Borenstein [Borentein,1991a] ou HIMM. Assim, ele se tornou a escolha para desenvolvimento e continuação do trabalho.

No que se refere a mapa de células (4.3.1) deve-se repetir que em geral eles são formados por um mapa 2D onde as células são preenchidas como ocupadas, vazias ou não exploradas. A forma de preenchimento e atualização desta células é que faz com que os métodos, (Capítulo 5) sejam diferentes entre si.

Na sua essência o HIMM é baseado em um outro método chamado VFH [Borentein,1991], também criado por Borenstein e Koren, o qual foi melhorado em relação à velocidade de construção de mapas. Por sua vez VFH é um aperfeiçoamento de um método chamado VFF [Borenstein,1992], feito por Borenstein e Koren. No VFH os autores criaram um filtro (que varia de peso de acordo com a complexidade do ambiente), chamado estágios de redução, que melhora a qualidade da construção de mapas. Por fim, o VFF ,base mais baixa do HIMM, é uma soma de dois métodos já existentes: o *Potential Fields* [Khatib,1986], que é advindo da teoria dos Campos Potenciais da mecânica dos fluidos, e um método criado no *Carnegie-Mellon University*, o *Evidence Grids* [Moravec,1996] para permitir que se crie um mapa.

Acompanhando a evolução do método Campos Potenciais até o HIMM, Figura 22, nota-se que a melhoria sempre ocorre no que se refere a construção de mapas.

Figura 22 – Árvore Genealógica da relação entre HIMM e Campos Potenciais.

5.5 A hierarquia das tarefas.

É natural acreditar que o método dos Campos Potenciais, sendo a base para o HIMM, execute uma tarefa de menor valor. Entretanto, tal crença não é verdade, pois o que torna um método de navegação eficiente, na maioria dos casos, é a sua capacidade de *desviar de obstáculos*, que normalmente é uma tarefa de nível mais baixo, Figura 23, e quase sempre reativa [Brooks, 1986]

Assim, o que torna o HIMM tão eficaz é na verdade o método que usa para tomar atitudes diante de obstáculos, ou seja, o método de Campos Potenciais. Ao contrário dos outros métodos com os quais foi comparado, HIMM não preenche um mapa com o objetivo de *localização*, mas

sim de *desvio de obstáculos* e *planejamento local* [Wijk,2001]. Essas duas tarefas também podem ser realizadas pelo método Campos Potenciais sozinho [Lazea,1997].

Figura 23 – Relação entre os tipos de tarefas realizadas por robôs móveis

5.6 Por que Campos Potenciais e não HIMM ?

Dentre os métodos de navegação aqui estudados, ficou claro a supremacia do método HIMM. O objetivo final deste trabalho era implementar, em FPGA (vide Capítulo 7), o método mais eficiente. Entretanto, se fosse implementado um método tão específico quanto o HIMM haveria uma limitação do o futuro robô, aliado ao fato de que o seu código fonte não nos foi cedido. O método de Campos Potenciais tem uma abrangência muito maior, ou seja, é usado por

mais pesquisadores em mais situações e em mais de um instituto, incluindo no VFF e conseqüentemente no HIMM.

Além da questão abrangência, o método Campos Potenciais é responsável por boa parte do sucesso do HIMM, segundo os próprios autores [Borenstein,1991] [Borenstein,1992], e não perde em nada para ele com relação a hierarquia das tarefas (vide subtítulo anterior).

Dessa forma, o método implementado em FPGA foi o de Campos Potenciais usado para *desvio de obstáculos e planejamento de caminhos locais*, Figura 23. A seguir tem-se uma explanação sobre o método (vide 6) e um breve relato de alguns trabalhos, os quais usam hoje o método de Campos Potenciais (vide 6.1), para que se justifique a afirmativa de sua abrangência.

Capítulo 6 – Campos Potenciais

O paradigma baseado em células (vide 4.3.1) serve a dois tipos de tarefas, Figura 1, a construção de mapas ou a navegação por mapas já construídos. Em ambos o caso, é necessário um algoritmo que sirva para *desvio de obstáculos* e *planejamento de caminhos locais*, Figura 23, (5.5). Neste trabalho serão usados mapas de células já construídos e o algoritmo de Campos Potenciais com a finalidade de *planejamento de caminhos locais*. Na Figura 24, é ilustrada a colocação dessa abordagem dentro de um contexto tradicional[Jones,1999].

Figura 24 – Este projeto dentro do contexto tradicional

O método de Campos Potenciais, oriundo da mecânica dos fluidos, foi trazido para o mundo computacional em 1978 (Khatib and Le Maitre 1978) e reusado, pelos mesmos autores, em outros trabalhos posteriores [Khatib,1986]. Entretanto, em mais de um caso a literatura [Everett,1995] atribui a paternidade do método a Bruce Krogh [Krogh,1984] .

A idéia, de forma mais simples, é que os obstáculos exercem uma força repulsiva e o destino do robô uma força atrativa, Figura 25, sendo que ambas, neste trabalho, podem ser configuradas, (vide Tabela 4) dependendo do resultado que se deseja.

Figura 25 – Tipos de Campos Potenciais

O campo de força (atrativo ou repulsivo) pode ser comparado a um campo magnético ou gravitacional. Uma das vantagens do método de Campos Potencias é que não importa o tamanho da célula ela sempre será representada por uma unidade dentro de um vetor. Dessa forma, utilizar um mapa de células com células de 50 x 50m ou um mapa de pixels a representação é a mesma [Murphy, 2000]. Outra, vantagem, de uma arquitetura baseada em campos potenciais é a facilidade de visualização do comportamento global do robô, ainda na fase de projeto: a observação do campo resultante da combinação dos vários comportamentos permite prever com relativa facilidade o que o robô fará em cada situação [Ribeiro,2001].

Dentro deste conceito existem cinco tipos de Campos Potenciais ou primitivas que podem ser usados isoladamente ou em conjunto para construir outros campos mais complexos: Uniforme, perpendicular, atrativa, repulsiva e tangencial.

Campo Uniforme: neste campo o robô pode sentir a mesma força em qualquer área não importa onde quer que esteja. Este tipo de campo é freqüentemente usado para direcionar o robô para alguma tarefa do tipo: “siga a linha”.

Campo Perpendicular: aqui o robô é orientado perpendicularmente a partir de algum objeto como uma parede ou borda. O campo pode partir tanto do objeto como ser orientado a ele.

Campo Atrativo: Figura 25 a), o círculo (objeto) no centro do campo exerce atração sobre o robô. Onde quer que o robô esteja dentro desse campo ele poderá sentir a atração do objeto. Esse campo pode ser usado para representar comportamentos da natureza onde um ser é atraído por comida, luz ou um objetivo. Neste experimento esse campo é usado para determinar o objetivo do robô. A força de atração deste campo, neste caso, pode ser configurada, Tabela 4.

Campo Repulsivo: Figura 25 b), essa força é exatamente a oposta a anterior. É freqüentemente usada para representar um objeto ou algo que o robô deve evitar (buraco, risco etc). Nesse caso esta força é usada para representar os obstáculos e a repulsão que ela exerce também pode ser configurada, Tabela 4.

Campo Tangencial: Esse campo é tangencial a um objeto. Podem ser descritos como linhas radiais circulares formando atração ao centro. Essa força poder ser usada no sentido horário ou anti-horário e normalmente tem a finalidade de fazer com que o robô circule algo, investigue.

Os pesquisadores (vide 6.1) que trabalharam com o método são unânimes em dizer que:

1. É um método elegante, simples e flexível [Goldberg,1995];
2. Sua maior (e única relatada) fraqueza é o problema do *local-mínimo* [Latombe,1991].

Nos casos de objetos côncavos ou de formações côncavas (parede + objeto ou objeto + objeto) o robô guiado por este algoritmo pode ficar preso, uma vez que todas as saídas ao seu redor possuem o mesmo peso atrativo e repulsivo. Essa falha, no algoritmo, é conhecida como problema do *Local-mínimo*.

Um método tão antigo com tanto a seu favor e tão pouco contra deve ter aplicações das mais diversificadas possíveis. O capítulo que se segue tem por objetivo comprovar tal afirmação.

6.1 O método Campos Potenciais no mundo.

Em uma rápida revisão têm-se: O VFF, VFH, HIMM (vide 5.4), Navbelt e o NavChair (vide 2.2.3) que são trabalhos que usam o método de Campos Potenciais e dispensam maiores apresentações.

No Brasil, pelo menos, dois grupos de pesquisa em dois institutos diferentes já utilizam a teoria dos Campos Potenciais como o principal meio de navegação para robôs móveis.

O Primeiro é a escola Politécnica da Universidade de São Paulo é um dos institutos que vem se destacando em robótica no país e possui atualmente em andamento quatro projetos [REACT,2002]. Um deles, o projeto REACT, trata-se de criar um sistema de navegação para um robô móvel autônomo com rodas, Pioneer, que além dos sonares possui uma câmera, Figura 26. O Projeto REACT tem como uma de suas principais partes no sistema de navegação o método de Campos Potenciais. O Outro, a Universidade de Brasília (UNB) no departamento de Engenharia Mecânica um grupo de nome GRACO [GRACO,2002] tem como um de seus projetos um “Algoritmo de Navegação para Robôs Autônomos Móveis” totalmente baseado em Campos Potenciais.

Figura 26 – Pioneer do LTI - PCS/EPUSP

Em 1984 Bruce H. Krogh da Carnegie Mellon University [Krogh,1984] utilizou o método , sem entretanto fazer nenhuma referencia a Khatib, criando o que chamou de GPF um método com resultados eficientes. A diferença em relação ao método anterior é que este possui técnicas para se livrar do problema do *local mínimo*. O método foi testado usando uma implementação em Pascal sendo executado em um HP 200-38C e a qualidade e velocidade obtidas foram melhor que as esperadas para a época.

Em 2000, na China, Liu Chengqing [Liu,2000] realizou um trabalho bastante interessante que cria o conceito de *Obstáculo Virtual* para eliminar a maior falha das várias *funções potenciais* já tentadas; o *Local Mínimo*. Este trabalho faz referencia aos dois, anteriores, como sendo os pais do método e cita outros seis como bons exemplos bem como faz questão de deixar claro que todos apresentam o problema do *local-mínimo*.

A idéia do obstáculo virtual funciona da seguinte maneira: através de quatro passos o algoritmo detecta se realmente está em um *local mínimo* e qual o espaço de navegação permitido. Concluindo esta etapa, e usando as informações adquiridas, através de mais dois

passos, o algoritmo conduz o robô para fora do *local-mínimo*. O Algoritmo foi exaustivamente testado no simulador do robô Nomad 2000 para ambientes côncavos tanto curvos como retilíneos e a solução mostrou-se um sucesso.

Um campo que tem recebido um interesse crescente é o de robôs móveis como se fossem carros em rodovias, robôs não-holonômicos (normalmente retangulares e que possuem graus de liberdade de movimentos limitados), ou seja robôs que se movem em ambientes dinâmicos onde outros obstáculos (robôs ou não) do mesmo tamanho que ele também se movem. O método de Campos Potenciais tem se mostrado um dos mais eficientes nesta área [Goldberg,1995]. Usando uma combinação de dois métodos de Campos Potenciais Haussoun, França, [Hassoun,1992] propõe um método para desvio de obstáculos estáticos e dinâmicos de um veículo, em uma espécie de estrada. Usando uma mistura de Campos Potenciais numéricos e Campos Potenciais generalizados [Krogh,1984] o experimento realizado em uma máquina SPARC e usando a linguagem C, apresentou excelente resultados.

Ainda dentro da modalidade de robôs não-holonômicos, em 1994 Oriolo e De Luca [De Luca,1994], Itália, criaram uma idéia, deveras simples, porém, muito eficiente e de baixo custo computacional. Para evitar o problema do *local-mínimo* eles usaram uma tecnologia que também advém da mecânica dos fluidos *vortex fields* (Campos de vorticidade). Se um fluido está deslizando em uma superfície e encontra um obstáculo, se ele tiver força e velocidade o suficiente para tal, ele irá circundá-lo, isto é vorticidade. Partindo deste princípio o método exposto por este trabalho sugere que ao encontrar um obstáculo o robô além das forças de atração e repulsão naturais no método de Campos Potenciais, é criada aqui o *vortex fields*, que determina, com base em cálculos de vorticidade, por que lado o robô irá desviar do obstáculo se pela direita ou esquerda. Dessa forma, simples e eficiente, o método elimina o problema do *local-mínimo*. O método foi simulado usando o Matlab e testes matemáticos comprovaram sua eficiência.

Não bastasse todas essas aplicações envolvendo um único robô, o método de Campos Potenciais também tem sido aplicado a “comunidades” de robôs [Reif,1995]. Na University of Pennsylvania [Song,2002] algoritmo foi usado para que um grupo de robôs se auto-organizasse sem a necessidade de um controlador central. Cada robô tem seu Campo Repulsivo (vide anteriormente) como se fosse um obstáculo, Figura 27. Uma larga análise de estabilidade foi feita em várias situações distintas. A estabilidade do algoritmo e os resultados gerados levou os

autores a concluírem que esse algoritmo pode gerenciar até centenas de robôs. Partindo de uma idéia parecida Jonh H. Reif [Reif,1995], da Duke University Carolina do Norte, criou um conjunto de regras para grupos de robôs baseados em Campos Potenciais. Além dos robôs usarem o método para desviarem entre si eles o usam como forma de navegação conjunta para realização de tarefas típicas militares agindo como se fosse um exército. A definição de *leis de força* (campos de repulsão ou atração) entre os grupos faz com que ora mantenham-se unidos e em direção a algum objetivo, ora se espalhem em direções contrárias.

Figura 27 – Campos Potenciais utilizados em grupos de robôs.

Brasil, Estados Unidos, Itália, França e China são algumas amostras da abrangência e sucesso do método escolhido para implementação em um dispositivo FPGA. Assim, fica evidente a contribuição que este trabalho poderá gerar, não somente dentro do contexto do projeto ARMOSH. Mais adiante será apresentado o algoritmo de Campos Potenciais o que foi usado neste trabalho, e sua implementação em software e hardware.

6.2 A Obtenção do Código.

A intenção do trabalho, como descrito inicialmente, em momento algum foi de inventar um método ou adicionar algo a algum existente. Além disso, não fazia parte deste projeto (não a princípio) implementar o código fonte do algoritmo escolhido.

Implementa-los a partir das descrições realizadas por artigos e outros documentos do gênero é tarefa não trivial. Esses documentos (artigos, relatórios técnicos, etc..) visavam deixar claro a realização e os benefícios conseguidos e não a implementação de um código na íntegra. Assim, implementar o código, seria tarefa digna de um mestrado, inteiro. Afinal, não existe disponível, de forma pública, apesar da sua popularidade, um código fonte do algoritmo de Campos Potenciais, em qualquer que seja a linguagem, pronto para ser utilizado por pesquisadores que desejam iniciar um trabalho a partir de algo já existente.

Tendo em mente as dificuldades de implementação e já que esse não era o objetivo, tinha-se a intenção, de que se conseguisse junto aos pesquisadores os códigos fontes utilizados em suas pesquisas e testes (fato esse que foi citado na qualificação).

Não foi assim tão simples. Muitos nem sequer responderam ao contato enquanto outros se deram ao trabalho de dizer “...perdi o código fonte...” . Alguns forneceram um emaranhado de código, incompleto, que não compilava em nenhum compilador conhecido, e mais tarde negaram-se a qualquer apoio para compilação ou entendimento do mesmo. Se implementar não estava no planejamento (tempo) e conseguir o código estava difícil, quase não houve uma saída.

6.3 O estudo de um caso.

Ainda assim, algumas, opções nos sobraram. Das opções oferecidas (códigos que nos foram gentilmente cedidos) a que mais nos pareceu útil foi a implementada pelo grupo GRACO [GRACO,2002] da UNB no trabalho “Algoritmos de Navegação para Robôs Autônomos Móveis”.

O trabalho foi orientado pelo Prof. Dr. Alberto J. Alvares e executado pelo aluno do Curso de Engenharia Mecânica Sérgio Gonsalves Tourino, o qual gentilmente não somente nos cedeu o código em Matlab, como disponibilizou-se para contatos e esclarecimentos. Esse trabalho despertou interesse por nossa parte pelo mesmo motivo que nos levou às outras escolhas. Durante toda a duração do trabalho procurou-se fazer escolhas que fossem altamente genéricas e que pudessem ser úteis ao projeto ARMOSH. O trabalho de Sérgio Gonsalves Tourino (UNB) esta ajudando a esse propósito e o algoritmo utilizado servirá como parte central (*Kernel*) para outros trabalhos [Fernandes,2001] [Scatena,2002] [Rodrigues,2002]

Depois de obtido o código, estudou-se possíveis melhorias. O trabalho todo era composto de 17 (dezesete) fontes em Matlab, dos quais, depois de entendido onde estava o que nos interessava, foram aproveitados 08 (oito) módulos *.m, 03 (três) *.txt e 01 (um) foi criado por este mestrando, a fim de facilitar os trabalhos de validação, vide Tabela 5. A definição de cada módulo, Tabela 3, e cada variável foi especificada mediante estudo e entendimento realizado por este. Os fontes em Matlab de cada módulo e os arquivos textos utilizados nesses testes podem ser encontrados no Apêndice A. Testes exaustivos para comprovar que a implementação, do método de Campos Potenciais, escolhida era de fato consistente foram realizados. Depois de entendido e testado em Matlab o código foi portado para C e novamente testado. A seguir o código foi minuciosamente retalhado e seguido por um estudo de performance. Cada laço foi analisado a fim de descobrir de que forma ele influenciava no conjunto.

Tabela 3 – Módulos e suas especificações

Seqüência	Nome do Arquivo fonte	Descrição
1	allrobot.m	Procedimento principal que executa o menu inicial e <i>chama</i> todos os outros procedimentos. Esse procedimento permite ao usuário distribuir obstáculos pelo ambiente.
2	placeobs.m	Grava em no arquivo obst.txt as coordenadas x,y a vorticidade e o raio de cada um dos obstáculos criados pelo usuário.
3	le_dados.m	Le dos arquivos de texto (orig.txt, obst.txt, dest.txt) todos os dados necessários para os calculos de percurso.
4	grupos.m	Agrupar os obstáculos que estão por demasia próximos e por onde o robô não poderia passar evitando cálculos desnecessários.
5	centro.m	Calcula o centroide dos grupos
6	ordena.m	Ordena em ordem crescente os grupos partindo do mais próximo da origem.

7	circle.m	Desenha os obstáculos na tela
8	robo2.m	Efetua o planejamento e trajetória propriamente ditos
9	obst.txt	Armazena uma matriz de 5 colunas por X linhas (quantos forem os obstáculos). As duas primeiras colunas são x,y, a terceira é f, a quarta é v e a quinta é o raio de cada obstáculo. (x,y,f e v são descritas na Tabela 6)
10	dest.txt	Armazena nas duas primeiras colunas as coordenadas x,y de destino, na terceira S e na quarta D (x,y,S e D são descritas na Tabela 6)
11	orig.txt	Armazena nas duas primeiras colunas as coordenadas x,y de origem, na terceira Rr e na quarta d (x,y,Rr e d são descritas na Tabela 6)
12	learquivo.m	Criado, por este mestrando, no decorrer do trabalho a fim de facilitar testes e comparações, este módulo lê os dados de um obst.txt que tenha sido definido fora da interface.

6.4 O estudo dos módulos.

O módulo principal allrobot.m chama todos os outros módulos. Em Matlab um módulo *.m é tratado como se fosse uma função. Assim o usuário executa o módulo allrobot.m o qual gera um menu e através deste os outros arquivos podem ser executados. A Figura 28 visa ilustrar como a comunicação entre os módulos funciona. As setas, quando ligadas entre os módulos, representam as chamadas, de um módulo a outro, sendo que o módulo que recebe a seta é o que foi chamado. No que se refere a arquivos texto a chegada da seta, partindo de algum módulo, representa que o módulo está escrevendo dados, o contrario representa leitura.

Figura 28 - Diagrama de comunicação dos módulos em Matlab

Antes de iniciar as análises dos testes dos módulos e seus resultados em Matlab, cabe ressaltar que como está sendo usado Campos Potencias sobre um Mapa de Células a representação do mundo do robô é uma matriz 10 x 10 metros esta matriz. Esta matriz deveria estar representada em todas as figuras, mas somente aparece na Figura 30 a), a fim de ilustrar como ficariam as outras. Entretanto, para melhor entendimento (das trajetórias), a matriz (grade) foi removida, o que não quer dizer que não exista.

Depois de entendido e reduzido o código aos oito arquivos que eram de interesse, tentou-se reduzi-lo ainda mais. Acreditou-se, em um primeiro momento, que os módulos *grupos.m*, *centro.m* e *ordena.m* não eram de grande valia.

Figura 29 – Testes realizados com os módulos

Alguns testes, iniciais, fizeram crer que eles não tinham nenhuma finalidade na maioria dos casos. Usamos uma versão sem as três funções para realizarmos os testes da Figura 29, decomposta em quatro outras. Note que com quatro ou quarenta e dois obstáculos o algoritmo se comporta bem. Entretanto quando se criam paredes, Figura 29 a), de obstáculos as cavidades deixadas entre os círculos conduz o robô ao problema do *local-mínimo*, Figura 29 b) (vide Capítulo 6).

Os três módulos juntos tentam uma solução para o problema de forma bem simples: os obstáculos que estão muito próximos têm sua variável de vorticidade (Tabela 6) v voltadas para o mesmo lado. Assim os obstáculos passam a formar um único objeto com um único campo de

repulsão o que faz com que o robô não tente entrar nas concavidades existentes. São módulos compostos de laços constantes e um custo computacional relativo.

Uma solução prática, para não usar os três módulos seria elevar f (força de repulsão da cada obstáculo) e diminuir S (força de atração do destino) equalizando, Figura 29 d), de forma que o robô não se aproxime tanto dos obstáculos não chegando dessa forma a encarar a concavidade que levaria ao *local-mínimo*. Usando-se deste artifício obtêm-se um bom resultado sem um grande custo computacional.

A desvantagem, deste artifício é que ao acrescentarmos um novo objeto ou ao adentrar uma nova sala, f e R teria que ser readequados. Como neste caso o robô não é usado para *Construção de Mapas e Navegação Simultâneos*, Figura 1), e sim para navegação em mapas previamente fornecidos seria conveniente que ao adentrar um ambiente novo ou receber uma nova tarefa terá receba junto com o mapa S e f calibrados.

6.5 O estudo do código.

A parte mais importante do algoritmo (laço principal) ocorre dentro do módulo robo2.m. Na Tabela 4, são documentadas a finalidade de cada uma das variáveis utilizada neste módulo, bem como onde foram definidas. Todas as variáveis em negrito foram definidas pelo usuário em

Figura 30 – Estudo dos módulos

arquivo texto (orig.txt, dest.txt e obst.txt), ou dentro dos fontes podendo ser posteriormente armazenadas em arquivos texto, ou ainda pelo quando o usuário interage com a interface.

Das variáveis, definidas pelo usuário, existem três que podem afetar de forma significativa os resultados obtidos pelo método: S , f e v .

Tabela 4 – Descrição das variáveis usadas no módulo *robo2.m*

Variável	Descrição	Definido em Texto	Definido em Interface	Definido no fonte
max_it	Número máximo de iterações			robo2.m
T	Tamanho da sala em metros (sala quadrada)			robo2.m
x,y	Coordenadas atuais do robô	orig.txt		
xd,yd	Coordenadas de destino do robô	dest.txt		
D	Distancia de parada antes da origem.	dest.txt		
S	Força de atração - exercida pelo destino	dest.txt		
No	Número de obstáculos		•	
xo,yo	Coordenadas de cada obstáculo	obst.txt	•	
F	Força de repulsão – exercida por cada obstáculo	obst.txt		placeobs.m
V	Vorticidade exercida por cada obstáculo	obst.txt		placeobs.m

Rr	Raio do Robô	orig.txt		
d	Distancia que o robô se locomove por iteração.	orig.txt		
R	Raio de cada obstáculo	obst.txt		placeobs.m
Td	Teta – ângulo do destino em relação ao robô – Atrativo			
Fx,fy	Somatória das forças atrativa e repulsivas convertidas em coordenadas para movimento do robô			
Ri	Distancia do obstáculo ao robô			
Ti	Teta – ângulo do obstáculo com relação ao robô – Repulsivo			
Fi	Teta final – ângulo obtido através de fx,fy para determinar a direção de movimento do robô.			

6.7 Calibração de variáveis

A variáveis S,f e v (explanadas na Tabela 4) devem ser definidas pelo usuário para cada caso. Os seus valores dependem da definição de outros valores como: O tamanho do ambiente T, o raio do robô Rr,o raio e a posição dos obstáculos r, e a origem e destino xd,yd.

Assim, em cada caso S,f e v devem ser testadas (calibradas) para que se encontre o melhor valor. A seguir tem demonstrar-se-á a reação do desempenho do código às mudanças realizadas nessas variáveis. Os valores, Tabela 5, usados são fictícios e foram escolhidos com base em um único objetivo: deixar mais visível as amostras dos testes que foram realizados.

Tabela 5 – Descrição dos valores utilizados nos testes.

Variável	Descrição	Valores utilizados
max_it	Número máximo de iterações	3000
T	Tamanho da sala em metros (sala quadrada)	10
xd,yd	Coordenadas de destino do robô	10,5
D	Distancia de parada antes da origem.	0,05
S	Força de atração - exercida pelo destino	Variante em cada exemplo
No	Número de obstáculos	12
xo,yo	Coordenadas de cada obstáculo	Vide apêndice A orig.txt
F	Força de repulsão – exercida por cada obstáculo	0.6
V	Vorticidade exercida por cada obstáculo	Aleatória
Rr	Raio do Robô	0.1
D	Distancia que o robô se locomove por iteração.	0.03
R	Raio de cada obstáculo	0,5

Os testes são iniciados pela observância do fator S. O fator S serve como a força de atração, que é exercida pelo destino, sobre o robô. A Figura 31 mostra um teste realizado com S crescente.

Note que um S muito pequeno, Figura 31 a), ocasiona duas falhas no algoritmo:

- a) A força de repulsão dos obstáculos, ainda que pequena (0,6) faz com que o robô crie sua rota muito longe dos obstáculos. Uma trajetória mínima (seria o esperado) não é alcançada.
- b) Mesmo quando já passou pelos obstáculos o robô segue incerto para o seu destino prefixado ($x=5, y=10$) que nem sequer chega a ser alcançado.

À medida que S é acrescido a trajetória em direção ao destino se torna mais direta e a distância com que o robô passa, em relação aos obstáculos, se torna menor, dessa forma, se houver continuidade do acréscimo de S, em um dado momento, o robô irá chocar-se com os obstáculos ou tentar atravessá-los, Figura 31 g). Concluindo, pode-se dizer que dependendo do que se pretende, que o S ideal está em algum momento entre $S=30$ e $S=200$, Figura 31 c) d) e) f). Entre $S=31$ e $S=50$, Figura 31 c) d) pode-se perceber uma pequena diferença que ocorre próximo ao obstáculo 14. Dada a atração mais forte em $S=50$ o robô quase tenta uma trajetória por entre os obstáculos ou rente a eles. A trajetória $S=100$, rente aos obstáculos, Figura 31 d) e), não é uma trajetória ótima, a não ser que a intenção seja justamente que o robô siga uma parede ou uma seqüência de obstáculos.

Figura 31 – Testes envolvendo o fator de Atração

A trajetória $S=200$ parece, para o caso desejado (uma trajetória mais curta), a mais interessante. A fim de evitar que o robô chegue tão próximo dos obstáculos pode-se, em qualquer dos exemplos, ir calibrando concomitantemente o fator de repulsão de cada obstáculo, f o qual foi usado fixo e igual para todos os obstáculos. Na Figura 32, dois testes são realizados em f , usando $S=200$ para ilustrar o que foi dito anteriormente.

Figura 32 – Testes envolvendo o fator de repulsão

Por fim, a última das variáveis definidas pelo usuário que pode afetar, significativamente, o desempenho do algoritmo é v . A variável v determina a vorticidade, para cada obstáculo, ou seja, se o robô irá se dirigir à direita ou à esquerda do obstáculo, supondo haver espaço livre dos dois lados é claro.

Não há muito o que se fazer em termos de testes de vorticidade. Ou opta-se por um v negativo (esquerda), Figura 32 a), ou v positivo (direita), Figura 32 b), ou um v ou deixa-se aleatório, Figura 32 c). Note que o robô pode, em uma determinada trajetória, receber influencia da vorticidade de um ou mais obstáculos assim mesmo que se determine v dependendo da distancia dos obstáculos pelo qual o robô terá que passar ele pode ser diferente do especificado.

Figura 32 – Teste envolvendo a vorticidade

Acredita-se que o v aleatório, Figura 32 c), é o mais próximo da realidade em um processo comum de locomoção e por isso optou-se por deixá-lo deste modo.

Capítulo 7 – Implementação do algoritmo de Campos Potencias em hardware reconfigurável.

A implementação do algoritmo de Campos Potenciais em hardware reconfigurável passou pelos seguintes passos:

- 1) Aprender a linguagem Matlab e entender o código em suas minúcias e detectar exatamente o que seria convertido em hardware;
- 2) Instalar, configurar e aprender o software Quartus II (vide 3.7);
- 3) Entender a comunicação entre a placa de desenvolvimento SOPC (Figura 14) e o Quartus II;
- 4) Entender o funcionamento do Processador Nios que deveria ser embutido na placa SOPC pelo software Quartus II;
- 5) Converter o Código Matlab para C Nios (uma linguagem C que tem suas particularidades) e executar o código dentro do processador Nios embutido em uma FPGA APEX 20 a qual fica alojada na placa SOPC (Figura 14);
- 6) Encontrar a parte do código de maior custo computacional (gargalo) para criar um hardware dedicado que pudesse executá-la (Figura 33);
- 7) Medir tempo, pulso de clock e se possível medir o desvio de precisão do algoritmo sendo executado em várias situações:
 - a. Em Matlab com um processador PIII 866 Mhz;
 - b. Em linguagem C, com um processador PIII 866 Mhz;
 - c. Em linguagem C, totalmente no processador Nios;

- d. Em C parte no processador Nios e parte em Hardware reconfigurável (Figura 33).

Sair do primeiro passo até o sexto foi tarefa razoavelmente trivial. Os dois passos seguintes serão descritos nos itens a seguir.

Figura 33 – Nios e hardware dedicado dentro de uma FPGA Apex.

7.3 Hardware dedicado

No decorrer da sexta tarefa (introdução do Capítulo 7) chegou-se a conclusão de que o trecho de maior custo computacional era na verdade apenas duas equações, que em linguagem C são elas:

$$(1) F_x = F_x + (-f * (\cos(t_i) * r_i) - v * \sin(t_i)) / (r_i^2) ;$$

$$(2) F_y = F_y + (-f * (\sin(t_i) * r_i) + v * \cos(t_i)) / (r_i^2) ;$$

Essas duas equações se situam dentro de um laço (for) e são realizadas enquanto o robô não chega ao destino, multiplicados pelo número de obstáculos existentes na sala, Figura 34. A cada passo que o robô se move as equações (1) e (2) são executadas X vezes, onde X é o número de obstáculos total existentes na sala. Ou seja o número total de execuções é dado pela equação:

$$(3) \text{ N}^\circ \text{ Total de execuções} = (\text{N}^\circ \text{ de passos percorridos da origem até o destino}) * (\text{N}^\circ \text{ Total de obstáculos na sala})$$

Além disto, se a sala estiver mapeada com uma célula de tamanho muito pequeno (para uma navegação mais precisa e desvio de objetos menores), este custo computacional será ainda mais alto.

```

while ((sqrt ( (x-xd)*(x-xd) + (y-yd)*(y-yd)) >D) & (i<max_it))
{
    .....
    .....
    for (o=0;o<no;o++)
    {
        .....
        .....
        Fx = Fx + (-f*(cos(ti)*ri) - v*sin(ti)) / (ri*ri) ;
        Fy = Fy + (-f*(sin(ti)*ri) + v*cos(ti)) / (ri*ri) ;
    }
    .....
    .....
}

```

Figura 34 – Trecho do código em linguagem em C

Ao término do laço as variáveis, Fx e Fy possuem a resultante do somatório das forças atrativas e repulsivas que darão o ângulo de movimento do robô. Converter apenas duas linhas de código fonte em hardware reconfigurável parece ser uma tarefa bastante trivial. Entretanto, não é assim tão simples. No código original todas as variáveis envolvidas nesse trecho são variáveis de ponto flutuante. Uma tentativa de arredondamento (testada em linguagem C) não

demonstrou sucesso. Além disto elas envolvem funções trigonométricas que também são difíceis de serem implementadas em hardware dedicado.

7.3.1 As Funções Trigonômétricas

O pequeno trecho de código que foi convertido em FPGA possuía duas barreiras. O primeira seria lidar com as funções trigonométricas e o segundo lidar com cálculos em pontos flutuantes (próximo item) .

Nas ferramentas EDA para FPGA não existem blocos prontos para nenhuma da função trigonométrica. Os Algoritmos existentes para implementação dessas funções em hardware (Capítulo 8) normalmente são baseados em tabelas. A grosso modo, o que é realizado nestes casos são uma série de comparações de acordo com a faixa de precisão necessária. Por exemplo, se a necessidade dos valores de Seno e Cosseno ocorrem de 5 ° em 5 ° então se toma 360° divide-se por 5 (72 posições) e converte-se em um vetor. Tem-se então 144 valores para cada ângulo desejado (com um intervalo aproximado de 5 a 5) ter-se-ia 72 comparações para somente então encontrar o par de valores desejados. Dessa forma, cada laço 72 comparações (realizadas em hardware) seriam necessárias. Tentativas dessa implementação (5 em 5) em C se mostraram ineficientes para a resolução do problema. Seria necessária uma precisão maior (dividir 360 em mais arcos) praticamente continua (valores de 1 em 1 ou menores) conseqüentemente; tabelas com mais valores e mais comparações. A perda de velocidade seria inevitável.

Usando Séries de Taylor [Giek,2001] tem-se que:

$$(3) \text{ Sen } x = \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} \dots$$

$$(4) \text{ Cos } x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \dots$$

A teoria diz que quanto mais termos da serie participarem do somatório, melhor a precisão alcançada em relação a função desejada. Entretanto, tinha-se que usar o mínimo possível para converter isso em hardware. Testes realizados mostraram que apenas o primeiro termo da função seno e os dois primeiros da função cosseno garantiam a precisão desejada para o exemplo (vide item 7.4).

Resolvida a questão trigonométrica, deparou-se com uma outra questão ainda de maior porte. Para todos os cálculos efetuados (mesmo usando os polinômios) são necessárias operações de ponto-flutuante. Tais operações não são triviais em hardware.

7.3.2 Aritmética de Ponto Flutuante

Todos os algoritmos de Campos Potenciais, analisados, utilizam ponto flutuante. Quando esses algoritmos são executados através de software não existem problemas, uma vez que, praticamente todas as linguagens de programação possuem as ferramentas necessárias para se representar e realizar operações com ponto flutuante. Quando se trata de hardware, entretanto a tarefa é bem mais complexa [Flynn,2001][Goldberg,1991].

A ferramenta Quartus II na sua última (2.1) versão traz a operação de multiplicação em, ponto flutuante, pronta para ser usada em uma FPGA como hardware dedicado. As outras três operações básicas envolvendo FPGAs ainda são motivo de pesquisa tanto da Altera quanto de pesquisadores[Rodrigues,2002]. Cabe salientar que o próprio processador Nios (processador que pode ser embutido na FPGA) não possui uma FPU (*Floating Point Unit*) e quando realiza esses cálculos o faz por software.

Dessa forma, tinha-se somente duas soluções, para este impasse:

- 1) Usar alguma técnica de normalização. Normalmente essas técnicas consistem em multiplicar os números reais (ponto-flutuante) por uma constante antes da entrada no hardware dedicado, fazendo com que o hardware trabalhe somente com inteiros. Quando do retorno do hardware ao processador, esse número é dividido pela mesma constante. Os ciclos de máquina crescem e, conseqüentemente, o custo computacional se torna maior, além do problema de convergência (perda de precisão).

2) Criar uma FPU, ou seja criar um hardware dedicado para realizar as quatro operações em ponto flutuante.

A segunda solução foi adotada. Em conjunto com um outro trabalho de mestrado [Rodrigues,2002], cujo o objetivo era a implementação de uma FPU em FPGA.

A implementação das operações de ponto-flutuante em hardware ocupam muito espaço físico dentro do dispositivo. Uma primeira versão com precisão de 32 bits não foi suportada pelo hardware (FPGA) utilizado. Uma segunda versão com precisão de 16 bits se mostrou ineficiente para a que o algoritmo atingisse seu objetivo, mesmo em uma situação muito simples (somente um obstáculo do tamanho do robô no meio de uma sala 30 x 30 m). Graças a exaustivos testes e tentativas contínuas, realizadas por este e pelo mestrando Murilo Inácio Rodrigues obteve-se uma versão de 32 bits, que além de ter o tamanho necessário para ser colocada em uma FPGA junto com o processador Nios, ainda resolvia o problema total da precisão, colaborando para que o algoritmo atingisse os resultados esperados.

Não bastassem os problemas de tamanho da FPGA e precisão que foram encontrados ainda foi necessário preocupar-se com o sincronismo entre o hardware dedicado (FPU) o Processador Nios. A cada vez que o Nios enviava instruções para a FPU era necessário que se esperasse algum tempo (gerando algumas instruções NOPs) para que se coletasse o resultado do cálculo. Cabe ressaltar que esse tipo de espera aumenta número de pulsos de clock (velocidade). Um estudo detalhado no código fonte em linguagem C Nios levou a otimização do código, de forma que se conseguisse minimizar esse número de clocks, Figuras 35 e 36, de aproximadamente seis milhões para aproximadamente três milhões.

Figura 35 – Telas com os pulsos de clocks das versões com otimização executadas no Nios $\cong$ 6.602.596 clocks

Figura 36 – Telas com os pulsos de clocks das versões com otimização executadas no Nios $\cong$ 2685 clocks

7.3.3 O Teste dos Polinômios

Houve uma grande preocupação se os polinômios utilizados responderiam bem a qualquer situação. Matematicamente, Figura 37, aproximadamente entre 120° e -120° a função representada pelo polinômio e a função seno estão muito próximas. Entretanto, não havia a certeza de que isto permitiria que o algoritmo convergisse. Com a função cosseno, ocorreu o mesmo problema, entretanto, a precisão nesse caso torna-se ainda maior atingindo valores aceitáveis entre -180° e 180° , Figura 38.

Figura 37 - Relação entre o Polinômio utilizado e a função seno

Figura 38 - Relação entre o Polinômio utilizado e a função cosseno

Testes foram realizados em situações em que o robô teria que seguir uma parede de obstáculos por um determinado período. Esses testes em outras ocasiões (Capítulo 6) se

mostraram os que mais exigiam do algoritmo. O algoritmo se comporta perfeitamente igual para ambos os casos, mesmo usando os polinômios para seno e cosseno, Figura 39.

Figura 39 – Comparação entre o uso das funções funções trigonométricas e o uso de polinômios.

7.5 O Exemplo implementado

No início da implementação detectou-se que, muito embora o processador Softcore Nios possuísse suporte a ponto-flutuante (emulado via software), em casos que se exigia demais dessa situação ocorria uma falha no sistema (reset).

Dessa forma, iniciaram-se os testes com um exemplo, Tabela 6, pequeno já que a intenção não era provar a eficiência do algoritmo de Campos Potencias, mas sim verificar suas condições de ser executado em uma FPGA. Os valores foram testados em Matlab, Figura 40, linguagem C Borland e posteriormente em linguagem C Nios sendo executado totalmente em Nios (sem o hardware dedicado FPU), vide resultados Tabela 7.

A implementação do código em linguagem C Nios para o uso de um hardware dedicado (FPU) requereu um esforço maior e contínuo de tentativas e tempo. O sincronismo entre o processador embutido Nios e a FPU não é natural (7.3.2).

Figura 40 – Trajetória, em matlab, do Exemplo implementado

Tabela 6 – Descrição dos valores utilizados na implementação.

Variável	Descrição	Valores utilizados
max_it	Número máximo de iterações	1000
T	Tamanho da sala em metros (sala quadrada)	30
xd,yd	Coordenadas de destino do robô	30,15
D	Distancia de parada antes da origem.	1
S	Força de atração - exercida pelo destino	30
No	Número de obstáculos	4
xo,yo	Coordenadas de cada obstáculo	10,15 25,15 16,22 16,06
F	Força de repulsão – exercida por cada obstáculo	15
V	Vorticidade exercida por cada obstáculo	Aleatória
Rr	Raio do Robô	2
D	Distancia que o robô se locomove por iteração.	1
R	Raio de cada obstáculo	2

7.6 As medições

Medir os resultados das implementações talvez tenha sido uma tarefa tão árdua quando a própria implementação.

O Matlab não oferece serviços de interrupção e as formas de medidas de tempo não são confiáveis. Como esta foi uma ferramenta usada apenas para validar o algoritmo (de forma visual

inclusive) preferiu-se não medi-las. As medidas conseguidas através da função Clock do compilador Borland, bem como as realizadas através da interrupção de Bios 1 C, mostraram-se duvidosas quando testadas em máquinas diferentes.

Standard 32-bit Nios system:
Version 2.0

Figura 41 – Blocos do Trabalho Final – Nios + FPU + Contador de pulsos.

Por fim, apesar das várias tentativas de medições confiáveis nas outras implementações, entendeu-se que as medidas realizadas no processador Nios comparadas entre suas diferentes implementações (com funções trigonométricas, com polinômios e com hardware dedicado), seriam as de real importância. Para esta tarefa usou-se um contador (Clock) da ferramenta Quartus ligado ao Nios e a FPU, Figura 41. O Nios possui uma função para contagem de tempo a qual também não funciona com perfeição, mesmo depois de ter-se tentado ajuda da altera. As medidas demonstradas foram realizadas através da execução de vários laços e depois calculadas. O processo foi repetido por três vezes quando se obtinham valores próximos e até por cinco vezes quando os valores diferenciavam um pouco mais. Apesar da falta de medidas, precisas, conclusões importantes podem ser observadas:

1) Os resultados obtidos pelo polinômio se mostraram surpreendentes tanto na precisão quanto nos tempo e ou ciclos de clock. No caso da implementação em Nios, por exemplo, a

execução completa do algoritmo, além de não causa choque com nenhum obstáculo, é mais rápida do que um trecho (um passo) da execução envolvendo as funções trigonométricas.

2) O A solução usando FPU, desenvolvida neste trabalho, mostrou-se um sucesso irrefutável. Tanto em tempo como em clock e precisão. Apesar de levar três passos a mais que a média ($41-38=3$) para chegar ao destino, ela não passa sobre os obstáculos o que demonstra uma precisão apurada.

3) As soluções de nº 2 e 5 as quais passaram por sobre os obstáculos serviram para provar a flexibilidade do algoritmo. Depois de adequados o fator S (Tabela 4) de atração do destino o algoritmo convergiu de forma aceitável como os outros.

4) No que se refere a tempo a solução implementada (nº 7) ficou muito aquém da solução em processador de uso geral. Entretanto, há que se observar a frequência de trabalho de ambos os processadores (866 Mghz e 33.3 Mghz).

Tabela 7 – Resultados obtidos nas diferentes implementações.

Nº	Tipo de Implementação	Ciclos de Clock - execução Completa	Tempo para execução Completa	Passos	Existe Choque	Ciclos de Clock em um Passo	Processador
1	Código em Matlab (Sen e Cos)	-	-	38	Não	-	PIII 866
2	Código em Matlab Com Polinômios	-	-	36*	Sim	-	PIII 866
				39	S=26		
3	Código em C (Sen e Cos)	-	0.8 Milisegundos	39	Não	-	PIII 866
4	Código em C Com Polinômios	-	0.8 Milisegundos	38	Não	-	PIII 866
5	Código em C (Sen e Cos) Nios	-	940 Milisegundos	36*	Sim	144.036	Nios 33.3
				42	S=26		
6	Código em C Com Polinômios Nios	137.561	-	38	Não	29.537	Nios 33.3
7	Polinômio rodando em Hardware dedicado + Resto do Código em C Nios	94.395	400 Milisegundos	41	Não	2.703	Nios 33.3
<ul style="list-style-type: none"> * As trajetórias com um número menor do que 38 passos indicam choque com obstáculo. Nesses casos o fator de atração, que era 30, foi reduzido para 26, a fim de evitar choques, e os passos obtidos podem ser observados logo abaixo do número com * 							

Em suma, a tabela e as conclusões obtidas através dela mostram que esse trabalho, não tão somente atingiu o objetivo proposto como também logrou êxito em dois fatores os quais não estavam totalmente previstos nos planos:

- 1) Realizou, a melhor escolha, de algoritmo para implementação (Campos Potenciais) dada a sua facilidade de implementação e adequação as mais adversas situações;
- 2) Conseguiu-se com o uso da FPU (hardware dedicado) o melhor tempo entre as possíveis implementações em FPGA (nº 5,6 e 7).

As tarefas que por ventura não puderam ser completadas neste trabalho por falta de recursos ou por não fazerem parte do objetivo deste são comentadas no Capítulo 8.

Capítulo 8 – Conclusão e Trabalhos Futuros

Este trabalho se iniciou com metas amplas: entender dentro do universo da robótica qual realmente seria um algoritmo útil e possível de ser convertido em hardware; esperar que alguém de simples e bom grado fornecesse esse código e ainda por cima ajudasse a entendê-lo; encontrar, decifrar, reduzir e quebrar em partes desse código.

Essas metas, foram convertidas em objetivos mais específicos: pesquisar para ter certeza o quanto o trabalho seria realmente útil; lidar com um processador que não possui, ainda sequer, as operações matemáticas básicas em ponto-flutuante; implementar, medir, comparar e por fim disponibilizar para futuros trabalhos uma explanação completa sobre navegação de robôs moveis, algoritmos de Campos Potenciais e hardware reconfigurável.

O resultado: uma fonte de referência inicial na área de navegação em robótica, literatura ainda rara em nossa língua; um código fonte de Campos Potenciais legível, claro e reutilizável em linguagens acessíveis e de reconhecimento mundial (linguagem C e Matlab); e sobretudo um hardware que deverá ser aproveitado não somente dentro do contexto ARMOSH como também para outros projeto de maior prospecção.

Como todo trabalho sempre termina como algo que pode e deve ser melhorado esse não é exceção.

8.1 Trabalhos Futuros

Embora ainda que de forma rápida pode-se ter contato com um algoritmo para implementação de funções trigonométricas em hardware, baseado em tabelas: CORDIC [Opencores, 2002]. O código fonte disponível no site em questão não funcionou dentro dos compiladores que o LCR dispunha e a conversão para que funcione requer um conhecimento

razoável em linguagens de implementação de hardware (Verilog, VHDL ou HDL). Uma tentativa de implementação em C não demonstrou muito sucesso. Como os caminhos tentados trouxeram êxito essa é uma das tarefas que ficam a ser realizada ou seja explorar as potencialidades do algoritmo CORDIC em para resolução de funções trigonométricas.

A altera [Altera,2002] publicou notícias sobre um Megacore (bibliotecas que podem ser acopladas ao Quartus II) que utiliza uma tecnologia denominada NCO (Numerically Controlled Oscillators) que, supostamente, realizaria a mesma tarefa do algoritmo acima, resolução de funções trigonométricas, entretanto, de forma bem mais rápida e prática, segundo a propaganda. Depois de um longo tempo preenchendo cadastros e requerendo a licença para teste, tudo o que foi recebido foram explicações de como funcionaria. Averiguar essa disponibilidade e testar resultado com o Campos Potencias seriam de grande valia.

Fazer uso da computação reconfigurável para criar uma ferramenta de utilização de Campos Potencias para navegação e robótica, de forma que os fatores de ajuste (Tabelas 5) possam ser adequados em tempo real. Assim, a medida que se muda de ambiente ou necessidade os valores seriam reconfigurados pela ferramenta. Essa seria uma ótima contribuição.

Fazer uso das funções criadas pela FPU de forma a agrupa-las ou torná-la paralela para que venham a executar polinômios de função trigonométrica de forma otimizada. Rapidez e precisão poderiam ser conseguidos para um problema o qual ainda não tem uma solução trivial: funções trigonométricas em hardware dedicado.

O algoritmo de Campos Potencias se mostrou prático, portátil e capaz de solucionar os mais diversos tipos de problemas relacionados a navegação, não somente em mapas já construídos, como também na construção destes. O Matlab por sua vez mostrou-se uma ferramenta poderosa para a implementação de algoritmos científicos de forma geral. Aliar os dois no sentido de formular uma ferramenta de simulação (Shapira, SNNS, etc) e teste de Campos Potenciais a fim de adaptar as variáveis (Tabela 4) a um ambiente desejado seria um ótimo trabalho. O que já existe, deixado por este trabalho, serviria de ponto de partida. Enfim espera-se que ou por continuidade ou por uso do pronto este trabalho seja realmente útil ao LCR e LABIC.

[Altera,2002] Altera,"Programmable Logic Device Family: Altera Corporation" in URL:<http://search.altera.com/query.html?qt=NCO&col=corp>

[Barros, 1996] Barros, M. A., "Uma Metodologia de Projeto e Implementação de Operadores para Processamento Digital de Imagens em Tempo Real usando Field Programmable Gate Arrays (FPGA)", Anais do IX Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens SIBGRAPI'96, pp 297-304, Caxambú, Brasil, 1996.

[Bischoff ,2000] Bischoff, R., "Recent Advances in the Development of the Humanoid Service Robot HERMES." 3rd EUREL Workshop and Masterclass – European Advanced Robotics Systems Development. Salford, U.K., Vol. I, pp. 125-134, 2000.

[Borenstein,1985] Borenstein J., Koren, Y., "A Mobile Platform for Nursing Robots",IEEE Transactions on Industrial Electronics, Vol. 32, No. 2, pp. 158-165, 1985.

[Borenstein,1990] Borenstein J., "The NavBelt - A Computerized Multi-Sensor Travel Aid for Active Guidance of the Blind." Proc. of the Fifth Annual CSUN Conference on Technology and Persons With Disabilities, Los Angeles, California, March 21 - 24, 1990, pp. 107-116, 1990.

[Borenstein,1991] Borestein J., Koren, Y., "The Vector Field Histogram – fast obstacle avoidancefor mobile Robots. IEEE Journal of Robotics and automation,7(3):278-288,1991.

[Borenstein,1991a] Borenstein J., Koren, Y., "Histogramic In-motion Mapping for Mobile Robot Obstacle Avoidance." *IEEE Journal of Robotics and Automation*, Vol. 7, No. 4, 1991, pp. 535-539, 1991.

[Borenstein,1992] Borenstein J., Raschke, U., "Real-time Obstacle Avoidance for Non-Point Mobile Robots." *SME Transactions on Robotics Research*. Vol. 2, pp. 2.1-2.10, 1992.

[Borenstein,1998] Borenstein J., Koren, Y., "Fast Obstacle Avoidance for Mobile Robots." *Tenth Israel Convention on CAD/CAM and Robotics*, Tel-Aviv, Israel, December 27-29,1988.

[Borenstein,1999] Borenstein J., Levine, S. P., Bell, D. A., Jaros, L. A., Simpson, R. C., Koren, Y., "The NavChair Assistive Wheelchair Navigation System." IEEE Transactions on Rehabilitation

Engineering, Vol. 7, N0. 4, December 1999, pp. 443-451, 1999.

[Borghi,1995] Borghi, G., Brugali, D.”Autonomous map learning for a Multi Sensor Mobile Robot Using diktiometric representation and negotiation mechanism”.Proc. Of Int. conf on Advanced Robotics, 1995.

[Bowen,1999]Bowen, J.P., Jifeng H. ,”Hardware Compilation: Verification and Rapid-prototyping”, RUCS Technical Report, RUCS/1999/TR/012/A, Department of Computer Science, The University of Reading, UK, 1999.

[Braun,2001]Braun F.,Waldvogel M.”Layered Protocol Wrappers for Internet Packet Processing in Reconfigurable Hardware”, Hot Interconnects (HotI) 9, Stanford, CA, USA, 2001.

[Brooks,1986]Brooks, R. A., “A Robust Layered Control System for a Mobile Robot”, IEEE Journal of Robotics and Automation, RA-2, pp. 14-23, 1986.

[Burns,1997] Burns J.,Donlin,A.,Hogg, J., Singh, S., De Wit, M., “A Dynamic Reconfiguration Run-Time System”, Proc. 5th Annual IEEE Symposium on Custom Computing Machines, IEEE Computer Society Press, 1997.

[Cardoso,2000] Cardoso, J. M. P., Vestias, M. P., “Architectures and Compilers to Support Reconfigurable Computing”, www.acm.org/crossroads/xrds5-3/reconcept.html, January 2000.

[Carnegie,2002] The Robotics Institue of Carnegie Mellon University in URL: <http://www.ri.cmu.edu/>

[De Luca,1994]De Luca A., Oriolo G., "Local incremental planning for nonholonomic mobile robots", IEEE Int. Conf. on Robotics and Automation, San Diego, CA, pp. 104-110, 1994.

[Dedieu,1998] Dedieu E., Milán, J. D. R., “Eficient *Occupancy Grids* for Variable Resolutions Map Building”. Institute for Systems, Information and Safety, 1998.

[Dehon,1996]Dehon, A.”Reconfigurable Architectures for General-Purpose Computing”, Massachussetts Institute of Technology, Ph.D. Thesis, October, 1996.

[Duckett,1999]Duckett, T., Nehmzow, U. , “Exploration of Unkonown enviroments using a compass, topological map and neural network”, In: Proceedings of the 1999 IEEE International Symposium on

computational Intelligence in Robotics and Automation, 1999.

[Duckett,2000]Duckett, T., Saffiotti, A., “Building Globally Consistent Gridmaps From topologies”, Center for Applied Autonomous Sensor systems, 2000.

[Elfes, 1989] Elfes, A.,”Using *Occupancy Grids* for Móbile Robot Perception and Navigation”, IEEE Computer, June. 1989.

[Engelson,1992]Engelson, S. P., McDermott, D. V.,”Error Correction in Mobile Robot map Learning”. Proc of IEEE Int. conf on Robotics and Automation, 2555-2560, 1992.

[Everett,1995] Everett H.R,”Sensors for Mobile Robots – Theory and Application”, A K Peters Ltda, Massachusetts. USA, 1995.

[Fernandes,2001] Fernandes, L C.,”Implementação de Algoritmos de Localização para Robôs Móveis”, Monografia de Qualificação ICMC/USP,2001

[Flynn,2001]Flynn, M.J., Stuart, F. O.,”Advanced Comnputer Arithmetic Desgin”,A Wiley-Interscience Publication ,2001

[Fujitsu,2002]Fujitsu, “Fujitsu Web Page” in URL: <http://pr.fujitsu.com/en/news/2001/09/10.html>, acessado em Agosto de 2002.

[Giek,2001] Giek K.,”Manual de Formas Técnicas”, H E Mus,24º Edição, 2001.

[Giorgini,2002]Giorgini A. L.,”Implementação de um Controlador PID Digital para Robótica Baseado em Computação Reconfigurável”, Dissertação de Mestrado ICMC/USP,2002.

[Gokhale,1998]Gokhale M., Stone J. M., “NAPA C: Compiling for a Hybrid RISC/FPGA Architecture”, *Proc. IEEE Symp. FCCM’98*, 1998.

[Goldberg,1991] Goldberg D.,”What every Computer Scientist Should Know About Floating Point Arithmetic”, ACM Computing Surveys, Vol. 23, N° 1, 1991.

[Goldberg,1995] Goldberg K.,Halperin D.,Latombe J. C.,Wilson R.,”Algorithmic Foundation of Robotics”, A K Peters Ltda, Massachusetts. USA, 1991.

[Gonçalves,2002] Gonçalves A. V.,”Architect-R: Uma ferramenta para desenvolvimento de robôs Reconfiguráveis”, Dissertação de Mestrado ICMC/USP,2002.

[GRACO,2002]GRACO, “Pagina Universidade de Brasilia” in URL:, <http://www.graco.unb.br/>, acessado em Agosto de 2002.

[Hassoun,1992]Hassoun, M. , DemazeauY. , Laugier C. , ”Motion control for a car-like robot: Potential field and multi-agent approaches”, In *Proceedings of the 1992 International Conference on Intelligent Robots and Systems (IROS'92)*, Raleigh, 1992.

[Hauser,1997] Hauser, ”The Garp Architecture”, University of California at Berkeley Department of Electrical Engineering and Computer Sciencies, 1997.

[Honda,2002]Honda, “Honda Web Page” in URL: <http://www.hondacorporate.com/new/index.html?subsection=asimo> , acessado em Agosto de 2002.

[Jones,1999] Jones J. L, Seiger B. A., Flynn A. M.,”Mobile Robots: Inspiration to Implementation”,A K Peters Ltda, Massachusetts. USA, 1999.

[Kalman ,1960]Kalman R.E.,”A new approach to linear filtering and prediction problems”, Trans. ASME, J. Basic Engineering 82 35–45, 1960.

[Kawada,2002] Kawada, “Kawada Web Page” in URL: http://www.kawada.co.jp/ams/hrp-2/index_e.html, acessado em Agosto de 2002.

[Khatib,1986]Khatib O., “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots”,In. Proc. Of the Int. conf. On Robotics and Automation, pages 500-505.IEEE ,1986

[Kongmunvattana,1998]Kongmunvattana, A., Prabhas C.,”A FPGA-based Behavioral Control System for a Mobile Robot”, IEEE Asia-Pacific Conference on Circuits and Systems, Chiangmai, Thailand, 1998.

[Kortenkamp,1994] Kortenkamp, D., Weymouth, T.,”Topological Mapping for Mobile Robots using a Combination of sonar and vision sensing. In proceedings of the Twelfth National Conference on Artificial Intelligence (AAI94) ,1994.

- [Krogh,1984]Krogh B. H., “A generalized Potential Field Approach to obstacle avoidance control”, Proceedings of the International Robotics Research Conference, pp.1150-1156, 1984.
- [Krotkov, 1995] Krotkov E., Hebert M.,Simmons R.,“Stereo Perception and Dead Reckoning for a Prototype Lunar Rover”, Autonomous Robots, Vol. 2, No. 4, pp 313-331, 1995.
- [K-Team, 2002]K-team, “K-Team Web Page” in URL:<http://www.k-team.com/robots/koala/>,2002, Acessado em Agosto de 2002.
- [Kuipers,1991]Kuipers, B. J., Byun, Y.T.,“A Robot exploration and mapping Strategy Based on a semantic hierarchy of spatial representations”.Robotics and Autonomous Systems,8:47-63, 1991.
- [Labic,2002]Laboratório de Inteligência Computacional do ICMC – USP São Carlos :URL: <http://labic.icmsc.sc.usp.br/portugues/>,acessado em Agosto de 2002.
- [Laine,1998]Laine J., Novick D., Arroyo A.,"Development of an Autonomous Aerial Vehicle Research Platform," *1998 Florida Recent Advances in Robotics*, Melbourne, Florida, 26-27 March 1998, 115-118, 1998.
- [Latombe,1991] Latombe J.,”Robot Motion Planning”,Chapter 7, Kluwer Academica Publishers, Norwell,MA,1991.
- [Lazea,1997] Lazea Gh.,Lupu E.,"Aspects on Path Planning for Mobile Robots",Technical Report, University of Cluj-Napoca, Romania, 1997 in URL <http://www.cs.utcluj.ro/~emlupu/papers.html>.
- [Liu,2000]Liu C., Ang M. H. Jr., Krishnan H., Lim S.Y.,"Virtual Obstacle Concept for Local Minimum Recovery in Potential Field Based Navigation", *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 24-28, pp. 983-988, 2000.
- [Lysaght, 1993] Lysaght, P., Dunlop, J., ”Dynamic Reconfiguration of FPGAs”, in More FPGAs. Abingdon EE&CS Books, pp. 82-94, 1993.
- [Mataric,1992] Mataric, M.J.,“Integration of Representation into goal-driven behavior-based robots”. IEEE Int. conf on Robotics and Automation, 8:304-312,1992.
- [Montemerlo,2002] Montemerlo M., Pineau J., Roy N., Thrun S., Verma V., “Experiences with a

Mobile Robotic Guide for the Elderly”, Proceedings of the AAAI National Conference on Artificial Intelligence, 2002.

[Moravec,1985] Moravec, H. P., Elfes, A.,”High Resolution Maps from Wide Angle Sonar”. In Proceedings of the IEEE international conference on Robotics and automation, 116-121,1985.

[Moravec,1996] Moravec H.,Martin M.C.,”Evidence Grids”,Tech. report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, March, 1996.

[Murphy, 2000] Murphy, Robin R., “Introduction do AI Robotics” ,The MIT Press,2000.

[Oliveira,1999]Oliveira A.; Melo A.,”Dyno – Um Robot com Hardware Reconfigurável”,Revista Electronica e Tekecomunicções, 1999.

[Opencores,2002] “OpenCores.org” in URL: <http://www.opencores.org/>, acessado em outubro 2002.

[Razdan,1994] Razdan, R., Brace K., Smith M. D., “PRISC Software Acceleration Techniques”, In *Proceedings of the IEEE International Conference on Computer Design*, Oct. 1994, pp. 145-149,1994.

[REACT,2002]REACT, “Pagina da Escola Politenica” in URL: <http://www.lti.pcs.usp.br/robotics/react/>, acessado em Agosto de 2002.

[Reif,1995] Reif J., Wang. H., “*Social potential fields: A distributed behavioral control for autonoomous robots.*”, In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Algorithmic Foundations of Robotics (WAFR)*, pages 331-345. A. K. Peters, Ltd, Wellesley, MA. 1995.

[Ribeiro,2001] Ribeiro C. H. C.,Costa A. H.R.,Romero R.A.F.,”Robôs Móveis Inteligentes: Princípios e Técnicas”,em URL:www.lti.pcs.usp.br/robotics/, acessado em Setembro de 2002.

[Ribeiro,2002] Ribeiro A.A.L.,”Estudo da Reconfigurabilidade das FPGAs”, Dissertação de Mestrado ICMC/USP,2002.

[Rodrigues,2002] Rodrigues, M. I.,”Implementação de uma FPU para Hardware Reconfigurável”, Dissertação de Mestrado ICMC/USP,em conclusão 2003.

- [Romero,2000] Romero, R. A., "Aprendizado de Robôs Móveis via Software e Hardwre - ARMOSH", Projeto de Pesquisa do ICMC-USP, Março de 2000.
- [ROV,2002]ROV, "Remote Operate Vehicle", in :URL:<http://physical.com/deep6/>, acessado em Agosto de 2002
- [Saripalli,2002]Saripalli S.,Montgomery J.F., Sukhatme G.S,"Visually-guided Landing of an Autonomous Aerial Vehicle", IEEE Transactions on Robotics and Automation,2002.
- [Saxena, 2000] Saxena N. R., Fernandez-Gomez S., Huang W.J., Mitra S., Yu S.Y., McCluskey E. J., "Dependable Computing and Online Testing in Adaptive and Configurable Systems," *IEEE Design and Test*, pp. 29-41, 2000.
- [Scatena,2002] Scatena J. M., "Implementação de Mapas Topológicos em para navegação de Robôs Móveis usando hardware reconfigurável ", Dissertação de Mestrado ICMC/USP,2002.
- [Schiele,1994] Schiele, B., Crowley, J., "A Comparison of positon stimation Techniques using occupancy grids. Proc. Of the Int. IEEE conf. On Robotics and automation, 1628-1634, 1994.
- [Schultz,1998]Schultz, A. C.,Adams, W.,Yamauchi, B., Jones M., "Unifying Exploration, Localization, Navigation and Planning Through a common Representation", Navy Center for Applied research in Artificial Inteligence, 1998.
- [Song,2002] Song P, Kumar V., "A Potential Field Based Approach to Multi-Robot Manipulation", Proceedings of the 2002 IEEE International Conference on Robotics and Automation, Washington D.C., 2002.
- [Star,2001] Star, "Star Bridge Systems", *Overview of SBS's Reconfigurable Computing Technology*, in <http://www.starbridgessystems.com/tech-over.html> – Acessado em Janeiro de 2001.
- [Taylor,1996] Taylor M. B., "Design Decisions in the Implementation of a Raw Architecture Workstation", Dartmouth College, 1996.
- [Teixeira,2002]Teixeira M. A., "Técnicas de Reconfigurabilidade das FPGA's Da Família APEX Altera", Dissertação de Mestrado ICMC/USP,2002.

[Thrun,1996]Thrun, S., Bücken A., “Learning Maps for Indoor Mobile Robot Navigation, CMU Technical Report CMU-XS-96-121, 1996.

[Thrun,1998]Thrun S.,Bücken A., Burgard W., Fox D., Fröhlingshaus T., Hennig D., Hofmann T., Krell M., Schimdt T.,”Map learning and high-speed navigation in RHINO.”, In *Proceedings of the 14. Fachgespräch Autonome Mobile Systeme (AMS '98)*. Springer Verlag, 1998.

[Thrun,1998a]Thrun, S.,Burgard W.,Fox D.,”A probabilistic approach to concurrent mapping and localization for mobile Robots”.Autonomous Robots 5, 253-271.Joint Issue with Machine Learning, 31:29-53, 1998.

[Thrun,2000]Thrun S. , Beetz M., Bennewitz M., Burgard W., Cremers A. B., Dellaert F.,Fox D., Hähnel D., Rosenberg C., Roy N., Schulte J., Schulz. D. , “Probabilistic algorithms and the interactive museum tour-guide robot minerva.”, *International Journal of Robotics Research*, 19(11):972-999, 2000.

[Tourino,1999]Tourino,S.R.G.,Álvares,A.J., ”Navigation Algorithm for Autonomous Mobile Robots”, Congresso Nacional de Estudantes de Engenharia Mecânica, 1999.

[Ulrich, 1991] Ulrich, I., Nourbakhsh, I.,“Appearance-Based Place Recognition for Topological Localization”, IEEE International Conference on Robotics and Automation, San Francisco, CA, April 2000, pp. 1023-1029. Best Vision Paper Award, 1991.

[Ulrich,1997]Ulrich, I., Mondada, F., Nicoud, J. D., "Autonomous Vacuum Cleaner", Robotics and Autonomous Systems, Vol. 19, pp. 233-245, 1997.

[Villasenor,1997]Villasenor J.,Mangione-Smith W. H.,”*Configurable Computing*”, Scientific American, 1997.

[Weinmann ,1991] Weinmann U.,Kunzmann A.,Strohmeir U., "Evaluation of FPGA Architectures," , W. Moore and W. Luk, Eds.: Abingdon EE&CS Books, pp. 147-156, 1991.

[Wijk, 2001] Wijk, O.,”Triangulation Based Fusion of Sonar Data With Application in Mobile Robot Mapping And Localization”, Phd. Thesis of Royal Institute of Technology, 2001.

[Wijk,1998] Wijk O., Jensfelt P., Christensen H.,” Triangulation Based Fusion of Ultrasonic Sensor

Data”, ICRA'98,1998.

[Wirthlin,1995] Wirthlin M. J., Hutchings B. L.,“A Dynamic Instruction Set Computer”. In *Proceedings of the 4th IEEE Symposium on FPGAs for Custom Computing Machines (FCCM'95)*, Napa Valley, California, USA, April 19-21, pp. 99-107, 1995.

[Wolf,2002]Wolf, D. F.,”Projeto de uma Rede Neural Utilizada no Reconhecimento de Gestos de Robôs Móveis Utilizando-se Computação Reconfigurável ”,Dissertação de Mestrado ICMC/USP,2002.

[Yamauchi,1998]Yamauchi, B.,Schultz, A.,Adams, W., Graves K.,”Exploration and espatial Learning Research at NCARAI”, Navy Center for Applied research in Artificial Inteligence, 1998.

[Youngblood, 1999]Youngblood, G. M., Holder, B. L., Cook ,D. J.,”A Framework for Autonomous Mobile Robot Exploation and Map Learning through the use of Place-Centric Occupancy Grids”,Computer Science and Engineering Department, The University of Texas Arlington,1999.

[Yu,2000]Yu, S. Y., Saxena, N. R., McCluskey E. J., "ACS Implementation of A Robotic Control Algorithm with Fault Tolerant Capabilities," *IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 175-184, Napa Valley, California, Apr. 16-19, 2000.

Apêndice A

Codigos Fontes e dados (textos) utilizados em Matlab

Allrobot.m

% Chama todas as rotinas de trajetorias

```

load orig.txt;
load dest.txt;
load obst.txt;

m=0;

while m ~= 4
    m = menu ('Simulador de Trajetorias', 'Criar Obstaculos','Gerar Trajetoria','Ler Arquivo','Sair');
    if m == 1
        placeobs;
    end

    if m == 2
        ledados;
        %grupos;
        %centro;
        %ordena;
        robo2;
    end
    if m == 3
        learquivo;
    end
    if m == 4
        exit;
    end
end

close

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

grupos.m

% Geracao dos grupos de obstaculos

ig = 0;
igtemp = ig;
g = zeros(1,no);

% Calculo das distancias entre dois obstaculos

for idi=1:no,
    for j=idi+1:no,
        dist(idi,j)=sqrt((xo(idi)-xo(j))^2+(yo(idi)-yo(j))^2)-R(idi)-R(j);
        dist(j,idi)=dist(idi,j);
    end
end

```

```

    end
    dist(idi,idi)=0;
end

for idi=1:no-1,
    flag = 1;
    for j=1:idi-1,
        if dist(idi,j) <= 2*Rr
            flag = 0;
            igtemp= g(j);
        end
    end
end

for j=idi+1:no,
    if dist(idi,j) < 2*Rr
        if (flag==1) & (g(idi)==0) & (g(j)==0)
            ig = ig + 1;
            igtemp = ig;
        end
        g(idi) = igtemp;
        g(j) = igtemp;
    end
end
end

for idi=1:no,
    if g(idi)== 0
        ig = ig + 1;
        g(idi) = ig;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

robo2.m

clear i;
% Número máximo de iterações
max_it = 1000;
% Tamanho em metros do 'quarto'
T = 10;
% Execucao da trajetoria
i=1;
g = zeros(1,3);%para guardar a trajetoria

while ((sqrt ((x(i)-xd)^2 + (y(i)-yd)^2) > D) & (i < max_it)),

```

```

    td = atan2 (yd-y(i),xd-x(i));

    Fx = S * cos (td);
    Fy = S * sin (td);

    for o=1:no,
        % if v(o) == 1
        %   circle (xo(o),yo(o),R(o),'g');
        % else
        %   circle (xo(o),yo(o),R(o),'b');
        % end

    %   if rem(o,2) == 0
    %       v(o)=-1;
    %   else
    %       v(o)=1;
    %   end
    ri = sqrt ((yo(o)-y(i))^2+(xo(o)-x(i))^2) - R(o) - Rr; % Distancia entre o Robo e o Obstaculo.
    ti = atan2 (yo(o)-y(i),xo(o)-x(i)); %Teta (Td) REpulsivo

    Fx = Fx + (-f(o)*cos(ti)*ri - v(o)*sin(ti)) / (ri^2) ;
    Fy = Fy + (-f(o)*sin(ti)*ri + v(o)*cos(ti)) / (ri^2) ;

    %       Fx = Fx + (-cos(ti)*ri - sin(ti)) / (ri^2) ;
    %       Fy = Fy + (-sin(ti)*ri + cos(ti)) / (ri^2) ;
    end

    fi = atan2 (Fy,Fx);
    x(i+1) = x(i) + d * cos(fi);
    y(i+1) = y(i) + d * sin(fi);

    g(i,1)=i;
    g(i,2)=x(i);
    g(i,3)=y(i);

    i = i + 1;

    if rem(i,5) == 0
        circle (x(i),y(i),Rr,'r')
    end

end

```

```
disp ('Trajetoria gerada. Plotando...');
```

```
plot (x(1:i),y(1:i),'w');  
axis ([0 T 0 T]);
```

```
disp ('Distância total:')  
(i-1)
```

```
save guarda.txt g -ascii
```

```
////////////////////////////////////////////////////////////////
```

```
grupos.m
```

```
% Calcula o centroide dos grupos
```

```
for ic = 1:ig,  
    numx(ic) = 0;  
    numy(ic) = 0;  
    den(ic) = 0;  
end
```

```
for j = 1:no,  
    numx(g(j)) = numx(g(j)) + xo(j)*R(j);  
    numy(g(j)) = numy(g(j)) + yo(j)*R(j);  
    den(g(j)) = den(g(j)) + R(j);  
end
```

```
for ic = 1:ig,  
    xc(ic) = numx(ic) / den(ic);  
    yc(ic) = numy(ic) / den(ic);  
end
```

```
clear ic,j;
```

```
////////////////////////////////////////////////////////////////
```

```
ordena.m
```

```
% Determinacao dos sentidos dos vortices dos grupos
```

```
% Calculo da distancia dos grupos a origem
```

```
for io=1:ig,  
    dg(io) = sqrt ((xc(io)-x(1))^2+(yc(io)-y(1))^2);  
    ng(io) = io;  
end
```

```
% Ordenacao crescente dos grupos
```

```
for io=1:ig-1,
    for j=io+1:ig,
        if dg(io)>dg(j)
            temp=dg(io);
            dg(io)=dg(j);
            dg(j)=temp;
            temp=ng(io);
            ng(io)=ng(j);
            ng(j)=temp;
        end
    end
end
```

```
% Definicao dos sentidos dos vortices
```

```
for io=1:ig,
    eg = round(rand);
    for j=1:no,
        if g(j)==ng(io)
            v(j)=(-1)^eg;
        end
    end
end
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
ledados.m
```

```
% Limpeza das variaveis
```

```
clear obst,dest,orig;
```

```
% Leitura dos arquivos de dados
```

```
load obst.txt;
load dest.txt;
load orig.txt;
% Transformacao dos dados para variaveis internas
```

```
xo = obst (:,1);
yo = obst (:,2);
f = obst (:,3); % Fator de Repulsão Assim com S seria o fator de atração
v = obst (:,4); % -1 elevado a algo ramdonico, Vorticidade (virar a direita ou a esquerda
quando encontrar Obstaculos
R = obst (:,5);
```



```

xd = dest (:,1);
yd = dest (:,2);
S = dest (:,3); % Fator de Atração
D = dest (:,4); % Distancia de Parada antes do Destino

x(1) = orig (1,1);
y(1) = orig (1,2);
d = orig (1,3); % tamanho do passo
Rr = orig (1,4); % raio do robot

n = size (xo); % Leitura do numero de obstaculos
no = n(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

learquivo.m

% Le obstaculos já criado em obst.txt

T = 10; %tamanho da sala
R =0.6; %raio dos obstaculos
close % Geracao dos obstaculos
obst = zeros(1,5);

load obst.txt;

xo = obst (:,1);
yo = obst (:,2);
R = obst (:,5);
n = size (xo); % Leitura do numero de obstaculos

for o=1:n,
    axis ([0 T 0 T]);
    circle (obst(o,1),obst(o,2),obst(o,5),'b');
    text (obst(o,1),obst(o,2),num2str(o));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

placeobs.m

% Cria obstaculos para a simulacao do robo

T = 10; %tamanho da sala

```

```

R=0.6; %raio dos obstaculos

% Geracao dos obstaculos

close
obst = zeros(1,5);

o=1;

xob = 0;
yob = 0;
xa = 1;
ya = 1;

while (xob ~= xa) | (yob ~= ya)
    axis ([0 T 0 T]);
    xa = xob;
    ya = yob;
    [xob,yob]=ginput(1);           % x,y
    if (xob ~= xa) | (yob ~= ya)
        obst(o,1) = xob;
        obst(o,2) = yob;
        obst(o,3) = 50;
        obst(o,4) = (-1)^(round(rand)); % v
        obst(o,5) = R; % R
        circle (obst(o,1),obst(o,2),obst(o,5),'b');
        text (obst(o,1),obst(o,2),num2str(o));
    end
    o = o+1;
end

save obst.txt obst -ascii

```

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

```

obst.txt
      x              y              f      v              R
1.9815668e+000 1.5204678e+000 40 1.0000000e+000 0.6
2.3041475e+000 2.5730994e+000 40 1.0000000e+000 0.6
2.5576037e+000 3.5672515e+000 40 1.0000000e+000 0.6

```

```

2.8110599e+000 4.6198830e+000 40 1.0000000e+000 0.6
3.0875576e+000 5.7602339e+000 40 1.0000000e+000 0.6
3.3410138e+000 6.7251462e+000 40 1.0000000e+000 0.6
3.6175115e+000 7.6900585e+000 40 1.0000000e+000 0.6
4.2857143e+000 8.2163743e+000 40 1.0000000e+000 0.6

```

```

/////////////////////////////////////////////////////////////////

```

orig.txt

```

xo      yo      d      Rr
0.0      5.0      0.03    0.1

```

```

/////////////////////////////////////////////////////////////////

```

dest.txt

```

xd      yd      S      D
10.0     5.0     300     0.05

```

```

/////////////////////////////////////////////////////////////////

```

```

/////////////////////////////////////////////////////////////////

```

rosentemp.c

//esta é a versão com seno e cosseno e tempo e clock

```

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <time.h>

```

```

#define d 1      //distancia que o robo se locomove por iteraçãO-tamanho do passo
#define D 2      //distancia da parada antes do destino

```

```

// #define S 15    // força de atraçãO exercida pelo destino
// #define f 5     // força de repulsãO

```

```

#define S 30     // força de atraçãO exercida pelo destino
#define f 10     // força de repulsãO

```

```

#define no 4      //numero de obstaculos

```

```

#define max_it 1000 // Numero maximo de iteracoes
#define T 30 // Tamanho em metros do 'quarto'

#define Rr 2 //raio do robo
#define R 2 //raio do obstaculo

struct time t,t1;

void robo2 ();

void robo2 (unsigned int xo[3],unsigned int yo[3])
{
    int v;
    // unsigned int o,i=1,xd=30,yd=5;
    unsigned int o,i=1,xd=30,yd=15;

    // float ti,td,fi,ri,x=0,y=5,Fx,Fy,aux_sen,aux_cos;
    float ti,td,fi,ri,x=0,y=15,Fx,Fy,aux_sen,aux_cos;

    clrscr();

    while ((sqrt ( (x-xd)*(x-xd) + (y-yd)*(y-yd)) >D) & (i<max_it))
    {
        td = atan2 (yd-y,xd-x);
        Fx = S * cos (td);
        Fy = S * sin (td);
        for (o=0;o<no;o++)
        {
            ri = sqrt ( ((yo[o]-y)*(yo[o]-y)) +
                        ((xo[o]-x)*(xo[o]-x)) ) -R-Rr; // Distancia entre o Robo e o Obstaculo.
            ti = atan2 (yo[o]-y,xo[o]-x); // Teta (Td) REpulsivo
            v = (o%2==0)?1:-1;//ramdonico de f entre 1 e -1
            Fx = Fx + (-f*(cos(ti)*ri) - v*sin(ti)) / (ri*ri) ;
            Fy = Fy + (-f*(sin(ti)*ri) + v*cos(ti)) / (ri*ri) ;
        }
        fi = atan2 (Fy,Fx);
        x = x + d * cos(fi);
        y = y + d * sin(fi);
        i = i + 1;
        printf("%d)-- %e , %e\n",i,x,y);
    }
    printf ("Distancia total:%d\n",i-1*d);
}

void main(void)
{
    unsigned int xm[4],ym[4],i;

```

```

int o;
clock_t start,end;
float ciclos;

/* xm[0]=10;
xm[1]=16;
xm[2]=22;
ym[0]=5;
ym[1]=5;
ym[2]=5;*/

xm[0]=10;
ym[0]=15;
xm[1]=25;
ym[1]=19;
xm[2]=16;
ym[2]=22;
xm[3]=16;
ym[3]=6;

gettime(&t);
start = clock();
        robo2(xm,ym);
end = clock();
gettime(&t1);
ciclos=(float) (end - start)/18.2;
printf("Tempo de Execuo: %2d:%02d:%02d.%02d\n",
        t1.ti_hour - t.ti_hour,
        t1.ti_min - t.ti_min,
        t1.ti_sec - t.ti_sec,
        t1.ti_hund - t.ti_hund);
printf("Clock de Execuo: %f\n",ciclos*18.2);
getch();

}

/////////////////////////////////////////////////////////////////

ropotemp.c

//esta é a versão com polinomio e tempo e clock

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <time.h>

```

```

#define d 1          //distancia que o robo se locomove por itera#Eo-
tamanho do passo
#define D 1          //distancia da parada antes do destino

//#define S 15        // for#Ea de atra#Eo exercida pelo destino
//#define f 5         //for#Ea de repuls#Eo

#define S 30        // for#Ea de atra#Eo exercida pelo destino
#define f 10        //for#Ea de repuls#Eo

#define no 4         //numero de obstaculos
#define max_it 1000  // Numero maximo de iteracoes
#define T 30         // Tamanho em metros do 'quarto'

#define Rr 2         //raio do robo
#define R 2          //raio do obstaculo

struct time t,t1;

void robo2 ();

void robo2 (unsigned int xo[3],unsigned int yo[3])
{
    int v;
    // unsigned int o,i=1,xd=30,yd=5;
    unsigned int o,i=1,xd=30,yd=15;

    // float ti,td,fi,ri,x=0,y=5,Fx,Fy,aux_sen,aux_cos;
    float ti,td,fi,ri,x=0,y=15,Fx,Fy,aux_sen,aux_cos;

    clrscr();

    while ((sqrt ( (x-xd)*(x-xd) + (y-yd)*(y-yd)) >D) & (i<max_it))
    {
        td = atan2 (yd-y,xd-x);
        Fx = S * cos (td);
        Fy = S * sin (td);
        for (o=0;o<no;o++)
        {
            ri = sqrt ( ((yo[o]-y)*(yo[o]-y)) +
                        ((xo[o]-x)*(xo[o]-x)) ) -R-Rr;    // Distancia entre o
Robo e o Obstaculo.
            ti = atan2 (yo[o]-y,xo[o]-x); // Teta (Td) REpulsivo
            v = (o%2==0)?1:-1;//ramdonico de f entre 1 e -1
            aux_sen=ti-((ti*ti*ti)/6);
            aux_cos=1-((ti*ti)/2) + ((ti*ti*ti*ti)/24);
            Fx = Fx + (-f*(aux_cos*ri) - v*aux_sen) / (ri*ri) ;
            Fy = Fy + (-f*(aux_sen*ri) + v*aux_cos) / (ri*ri) ;
        }
        fi = atan2 (Fy,Fx);
        x = x + d * cos(fi);
        y = y + d * sin(fi);
        i = i + 1;
    }
}

```

```

        printf("%d)-- %e , %e\n",i,x,y);
    }
    printf ("Distancia total:%d\n",i-1*d);
}

void main(void)
{
    unsigned int xm[4],ym[4],i;
    int o;
    clock_t start,end;
    float ciclos;

    /*  xm[0]=10;
    xm[1]=16;
    xm[2]=22;
    ym[0]=5;
    ym[1]=5;
    ym[2]=5;*/

    xm[0]=10;
    ym[0]=15;
    xm[1]=25;
    ym[1]=15;
    xm[2]=16;
    ym[2]=22;
    xm[3]=16;
    ym[3]=6;

    gettimeofday(&t);
    start = clock();
        robo2(xm,ym);
    end = clock();
    gettimeofday(&t1);
    ciclos=(float) (end - start)/18.2;
    printf("Tempo de Execu#o: %2d:%02d:%02d.%02d\n",
        t1.ti_hour - t.ti_hour,
        t1.ti_min - t.ti_min,
        t1.ti_sec - t.ti_sec,
        t1.ti_hund - t.ti_hund);
    printf("Clock de Execu#o: %f\n",ciclos*18.2);
    getch();
}

```