Implementação do Método de Campos Potenciais para Navegação de Robôs Móveis Baseado em Computação Reconfigurável

ROVILSON MEZÊNCIO

Resumo: Este trabalho descreve o potencial do hardware reconfigurável tem se mostrado uma saída para o desenvolvimento de plataformas especificas.
Palavras-chave: FPGA, Robótica e Hardware.

Abstract: This paper describes an approach to convert the Potential Fields algortihm for mobile robots in reconfigurable hardware using FPGA.
Key-Works: FPGA, Robotic, Hardware.

Introdução

O hardware reconfigurável tem se mostrado uma saída para o desenvolvimento de plataformas especificas. 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 especificas para software que eram executados em processadores de uso geral [Oliveira,1999].

Aliado a essa idéia, observou-se que, os algoritmos de navegação baseados em células [Moravec,1996] ainda são de alto custo computacional [Wijk,2001]. Dentre os cincos algoritmos mais conhecidos mais conhecidos, mais conhecidos, o HIMM (Histogramic In Motion Mapping) [Borenstein,1991a] se provou o melhor [Wijk,2001]. Depois de uma análise 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 a implementação do algoritmo de Campos Potenciais em uma FPGA Altera.

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 (Triangulation Based Fusion)

Seu método é comparado a outros quatros 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, no trabalho de Olli, usando cada um dos cinco méto-

[1]Cristiane Pires Martins Aluna do Curso de Administração de Empresas, 4° Periodo Da Faculdade Delta.
[2]Daniele Lopes Oliveira Mestre em Ecologia e Produção Sustentável-UCG, Graduada em Direito-UCG, especialista em Docência Superior-Faculdade Lions e Professora da Faculdade Delta (danielelopes_oliveira@hotmail.com). UNIVERSIDADE CATÓLICA DE GOIÁS. Mestrado em Ecologia e Produção Sustentável. Campus II, Cx Postal 86. Av. Engler, Setor Jardim Mariliza, CEP: 74.605-010. Goiânia, Goiás. Brasil. (meps@ucg.br).

dos. Um ao longo de um corredor e outro em uma sala com mobília. 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.

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 1.

Partindo da Tabela 1 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.

Desempenho na Sala.

O tamanho da célula utilizada no experimento da sala foi de 0.1m. A Figura 1(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 freqüentemente 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

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

Tabela 1 – Comparação entre os diferentes métodos. O tempo é medido em segundos e o consumo de memória em OG, uma medida relativa criada pelo autor.

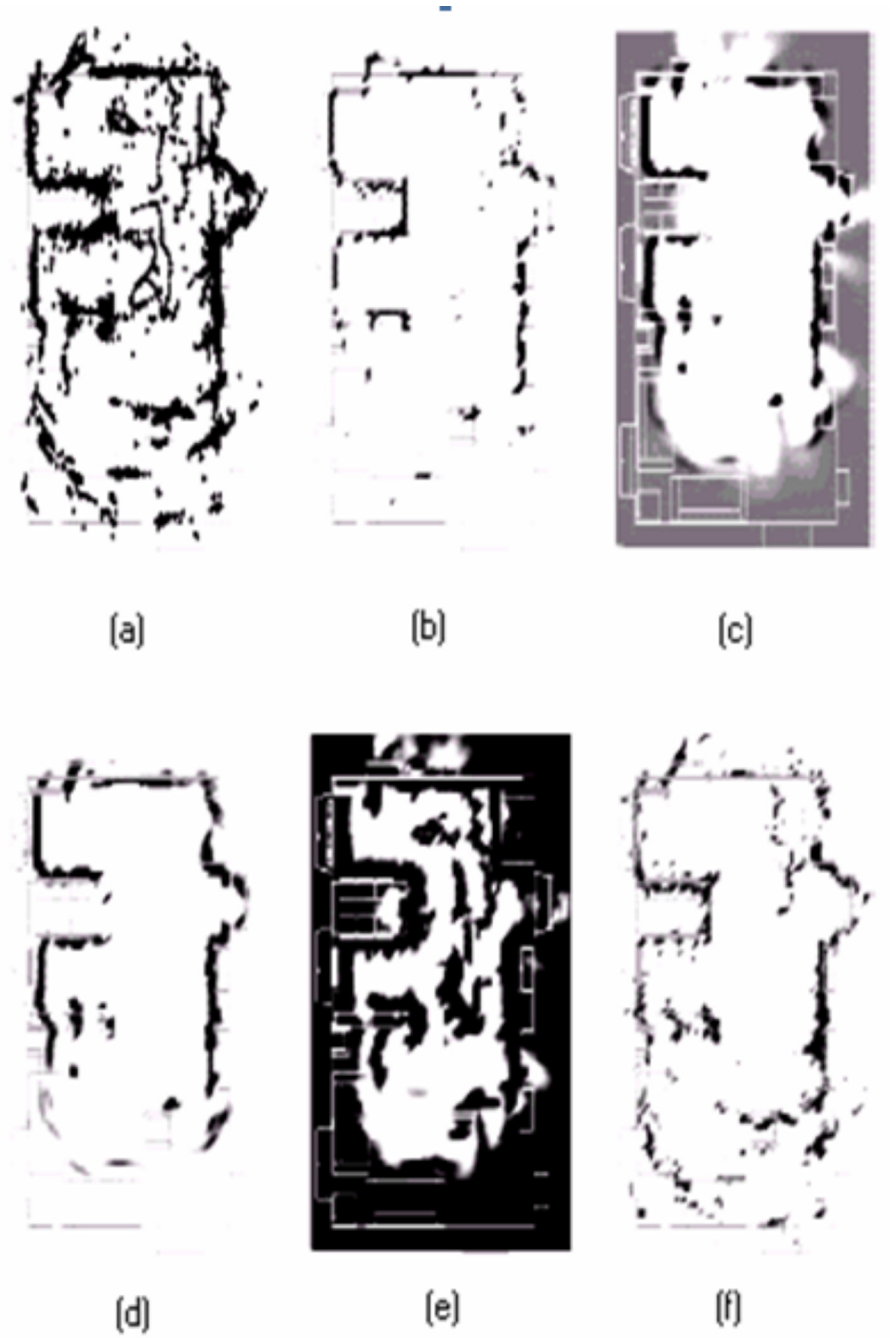

Figura 1 – a) Dados do sonar usados no experimento b) Mapa TBF c) Bayesiano d) Dempster e) Fuzzy f) Borenstein

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 1 (f) no canto superior direito.

No mapa com o método TBF, Wijk, Figura 1 (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 esparso, mas ainda assim nos leva a uma descrição do ambiente.

No mapa de células Bayesiano, Figura 1 (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 1 (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 1 (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 1 (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.

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 2 (a).

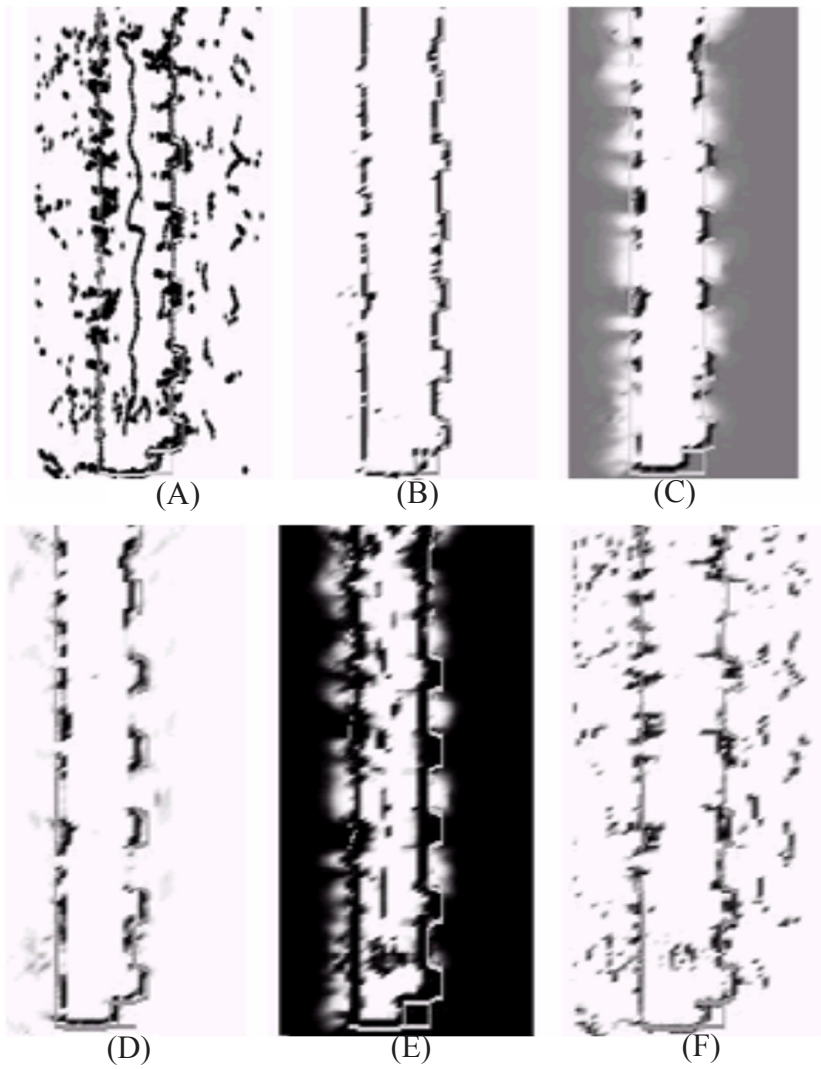

Figura 2. a) rastreamento realizado pelo sonar no experimento corredor b) TBF.c) Bayesiano d) Dempster-Shafer e) Fuzzy f) Borenstein HIMM

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.

A Figura 2 (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

2 (c). A partir desta imagem pode-se notar que muitas partes do corredor, sobretudo as próximas às portas, não foram bem captadas. Entretanto as passagem foram bem detectadas e o sinal da pessoa também foi totalmente removido.

Na Figura 2 (d) 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.

O mapa obtido pelo método Fuzzy pode ser observado na Figura 2 (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 2 (c). Este metódo, 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.

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 emcorredor 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 caraterísticas 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 muitos 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.

HIMM x Campos Potenciais. Qual a relação?

Na sua essência o HIMM é baseado em outro método chamado VFH (Vector Field Histogram) [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 (Vector Field Force)[Borenstein,1992], feito por Borenstein e Koren. No VFH os

autores criaram um filtro (que variade 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.

Dentre os métodos de navegação acompanhados, ficou claro a supremacia do método HIMM. Seria óbvio então implementa-lo em uma FPGA Field Programable Gate Array) 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.

Dessa forma, o método implementado em FPGA foi o de Campos Potenciais usado para desvio de obstáculos e planejamento de caminhos locais

Campos Potenciais

O paradigma baseado em células serve a dois tipos de tarefas: 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 locai. Neste trabalho serão usados mapas de células já construídos e o algoritmo de Campos Potenciais com a finalidade de planejamento de caminhos locais.

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 sendo que ambas, neste trabalho, podem ser configuradas dependendo do resultado que se deseja.

O campo de força (atrativo ou repulsivo) pode ser comparado a um campo magnético ou gravitacional. Uma das vantagens do método de Campos Potenciais 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: o 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.

Campo Repulsivo: 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.

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 que trabalharam com o método são unânimes

em dizer que:

É um método elegante, simples e flexível [Goldberg,1995];

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.

A Adequação de um Código

A intenção do trabalho 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, d e 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).

A maioria forneceu 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.

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 (Grupo de Automação e Controle) [GRACO,2002] da UNB (Universidade de Brasília) 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 ás 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 (dezessete) fontes em Matlab, dos quais, depois de entendido onde estava o que nos interessava, foram aproveitados 08 (oito) módulos Matlab, sendo inclusive incluído 01 (um) modulo para melhor estudo e entedimento.

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 [Mezencio,2002].

Implementação do algoritmo de Campo 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 (ferramenta EDA para uso de FPGA); Entender a comunicação entre a placa de desenvolvimento SOPC e o Quartus II; Entender o funcionamento do Processa-

dor Nios que deveria ser embutido na placa SOPC pelo software Quartus II; 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;

Encontrar a parte do código de maior custo computacional (gargalo) para criar um hardware dedicado que pudesse executá-la;

Medir tempo, pulso de clock e se possível medir o desvio de precisão do algoritmo sendo executado em várias situações:

Em Matlab com um processador PIII 866 Mhz;

Em linguagem C, com um processador PIII 866 Mhz;

Em linguagem C, totalmente no processador Nios; Em C parte no processador Nios e parte em Hardware reconfigurável (Figura 3).

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

Hardware dedicado

No decorrer da sexta tarefa 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) Fx = Fx + (-f*(cos(ti)*ri) - v*sin(ti)) / (ri^2) ;
(2) Fy = Fy + (-f*(sin(ti)*ri) + v*cos(ti)) / (ri^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. A cada passo que o robô se move as equações (1) e (2) são executadas X vezes, ode X é o número de obstáculos total existentes na sala.Ou seja o número total de execuções é dado pela equação:

*(3) Nº Total de execuções=(Nº de passos percorridos da origem até o destino) * (Nº 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.

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.

As Funções Trigonométricas

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 (Electronic Design Automation) para FPGA não existem blocos prontos para nenhuma da função trigonométrica. Os Algoritmos existentes para implementação dessas funções em hardware normalmente são baseados em tabelas. 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

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

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

[Giek,2001] tem-se que:

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.

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.

Aritmética de Ponto Flutuante

Todos os algoritmos de Campos Potenciais, analisados, uti-

lizam 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 das maquinas crescem e, constatimente o custo comportacional se torna maior, alem do problema de convergência.

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 verão 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). Após longas tentativas e teste em conjunto uma versão de 32 bits foi obtida. Essa 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 calculo. Cabe ressaltar que esse tipo de espera aumenta número de pulsos de clock (velocidade). Um estudo detalhado no código fonte em linguagem C Nios conduziu a uma otimização do código, de forma que se conseguisse minimizar esse número de clocks, de aproximadamente seis milhões para aproximadamente três milhões.

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 medí-las. As medidas conseguidas, Tabela 2, 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.

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. 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 surpre-

endentes 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, hardware dedicado mostrou-se um sucesso irrefutável.Tanto em tempo como em clock e precisão. Apesar de levar três passos a mais que a media (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 freqüência de trabalho de ambos os processadores (866 Mghz e 33.3 Mghz).

Em suma, a Tabela 2, 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 a seguir.

Conclusão

Este trabalho se iniciou com metas amplas: entender dentro do universo da robótica qual re-

Tabela 2 – Resultados obtidos nas diferentes implementações

Nº	Implementação	Ciclos de Clock Completa	Tempo para exec.	Passos	Ciclos Clock - um Passo	Proc
1	Código em Matlab (Sen e Cos)	-	-	38	-	PIII 866
2	Código em Matlab Com Polinômios	-	-	36* 39	-	PIII 866
3	Código em C (Sen e Cos)	-	0.8 Milisegundos	39	-	PIII 866
4	Código em C Com Polinômios	-	0.8 Milisegundos	38	-	PIII 866
5	Código em C (Sen e Cos) Nios	-	940 Milisegundos	36* 42	144.036	Nios 33.3
6	Código em C Com Polinômios Nios	137.561	-	38	29.537	Nios 33.3
7	Polinômio em Hardware dedicado + Resto do Código em C Nios	94.395	400 Milisegundos	41	2.703	Nios 33.3

** 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 **

almente 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 projetos.

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 (COordinate Rotation DIgital Computer) [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 explanações 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 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, portável 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 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 (Laboratório de Computação Reconfigurável) LABIC (Laboratóro de Inteligência Computacional) .

Referências Bibliográficas:

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

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

[Borenstein,1992] Borenstein J., Raschke, U., "Real- time Obstacle Avoidance for Non-Point-Mobile Robots." SME Transac-

tions on Robotics Research. Vol. 2, pp. 2.1-2.10,1992.

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

[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.

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

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

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

[Krogh,1984] Krogh B. H., “A generalized Potential Field Approach to obstacle avoidance control”, Procedings of the International Robotics Research Conference, pp.1150-1156, 1984.

[Latombe,1991] Latombe J., ”Robot Motion Planning”,Chapter 7, Kluwer Academica Publishers, Norwell,MA,1991.

[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.

[Mezencio, 2002] Mezencio, R.,” Implementação do método de campos potenciais para navegação de robôs móveis baseada em computação reconfigurável.”, Dissertação de Mestrado ICMC/USP,2002.

[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.

[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.

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

[Scatena,2002]ScatenaJ. M.,” Implementação de Mapas Topológicos em para navegação de Robôs Móveis usando hardware reconfigurável “, Dissertação de Mestrado ICMC/USP,2002.

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