Implementação de campos potenciais aplicado robótica utilizando ROS

Anderson Camargo Sant'Ana
PPGCC - Programa de Pós-Graduação em Ciência da Computação - PUCRS
anderson.santana.001@acad.pucrs.com

***Abstract*—O trabalho propõe a implementação de campos potenciais para traçar a trajetória de um robô móvel de forma que o robô não colida com os obstáculos presentes no ambiente. Esta técnica foi implementada na plataforma chamada ROS (Robot Operating System).**

***Keywords*—Campos potenciais, ROS, robô.**

I. Introdução

Este presente trabalho foi requisitado pela disciplina de Robótica Móvel Inteligente, e tem o objetivo de implementar um campo potencial para traçar a trajetória de um robô móvel. Em um contexto geral da disciplina, foi proposto uma divisão das atividades para os alunos cujo o objetivo geral proporcionar o entendimento das técnicas aplicadas a robótica para os alunos de pós-graduação.

II. Fundamentos

O planejamento de caminho do campo potencial consiste em criar um campo ou gradiente no mapa do robô que direciona o robô para a posição de destino a partir de várias posições de partida. O método de campo potencial trata o robô como um ponto sob a influência de um campo de potencial artificial. O robô se move acompanhando o campo [1].

A posição der destino (ocupa mínimo neste espaço no mapa) atua como uma fora atrativa para o robô, e os obstáculos atuam como picos ou forças repulsivas. Tal campo de potencial artificial suavemente guia o robô em direção ao destino enquanto simultaneamente evita obstáculos conhecidos [1].

É importante notar, porém, que isso mais do que apenas planejar o caminho. O campo resultante também é uma lei de controle do robô. Supondo que o robô possa localizar sua posição em relação ao mapa e ao campo em potencial, ele sempre pode determinar sua próxima ação necessária com base no campo.

A ideia básica por trás de todas as abordagens de campo em potencial que o robô é atraído para o destino, sendo repelido pelos obstáculos que são conhecidos de antem/ ao. Se novos obstáculos aparecerem durante o movimento do robô, pode-se atualizar o campo potencial para integrar essa nova informação [1].

As figuras 1,2 e 3, extraídas de [1] exemplificam a técnica de campos potenciais .

A Figura 1 mostra a mapa criado onde o robô trafegará. Este mapa pode ser criado através de uma varredura do ambiente utilizando alguns sensores, como por exemplo, sensores ultrassônicos, sensores ópticos que identifiquem os objetos presentes no ambiente e marquem no mapa.

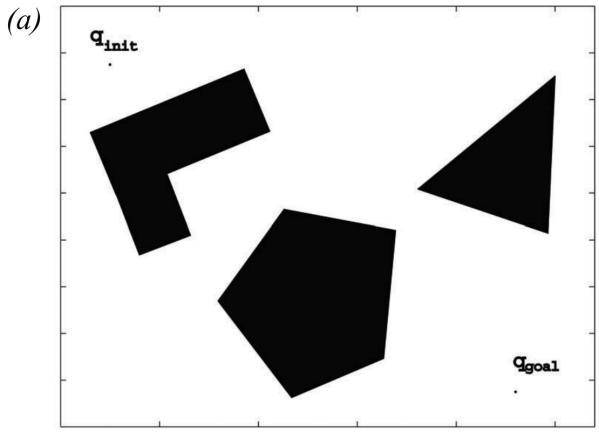

Fig. 1. Configuração dos obstáculos [1].

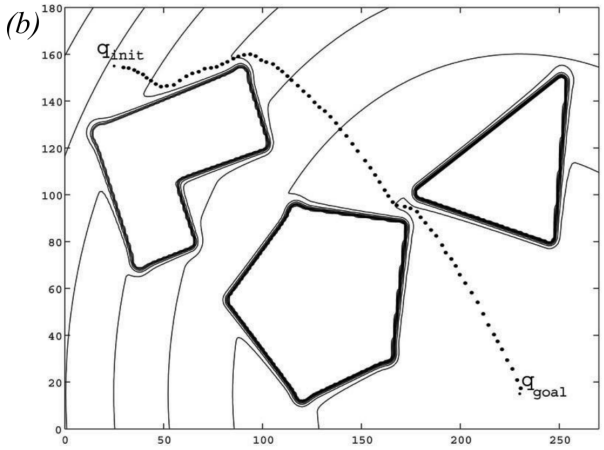

Fig. 2. Curvas de nível do campo potential e caminho seguido pelo robô [1].

A Figura 2 mostra a tragetória gerada pelos campos potenciais, nota-se que o robô desvia de todos os objetos presentes no mapa percurso mais direto ao destino.

A fig. 3 mostra os objetos como picos e o destino como um buraco, assim o robô converge para o destino proposto, para entender melhor, imagina-se que o robô tem o mesmo comportamento que uma bola que rolaria num morro.

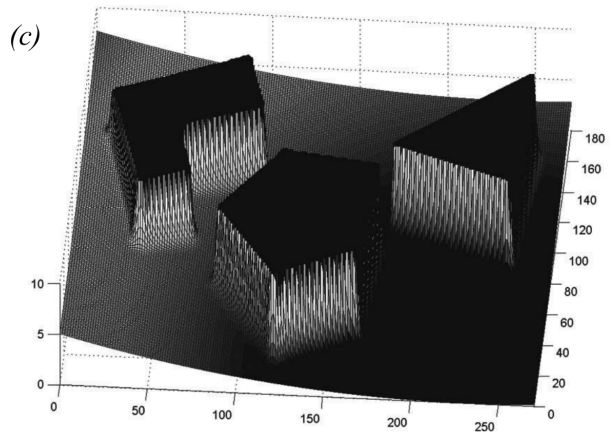

Fig. 3. Campo potencial [1].

III. Implementação do campo potencial

Nós implementamos a técnica de campos potenciais no ambiente ROS (*Robot Operation System*). Nós criamos um ambiente para o robô se deslocar. Este ambiente é possível ver no Gazebo, conforme mostra a figura 4. O robô utilizado é o TurtleBot disponível no ambiente ROS.

A. Cenário

Na Figura 4 mostra cinco objetos colocados aleatoriamente no ambiente, junto com uma barreira em forma de quadrado para limitar o espaço de tráfego do robô.

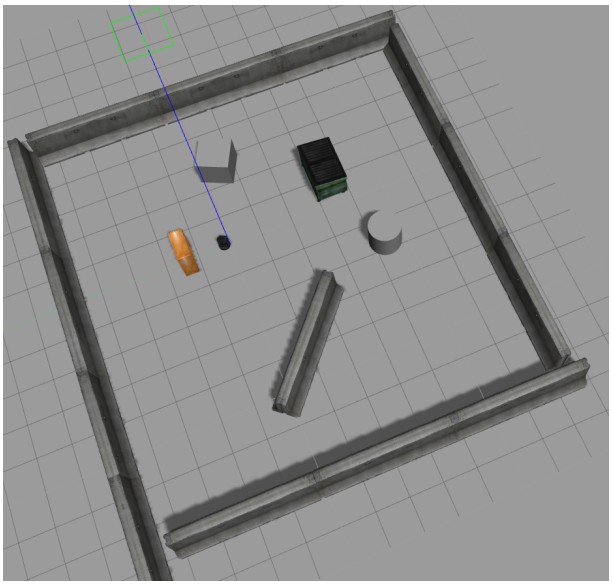

Fig. 4. Configuração dos obstáculos no ambiente Gazebo.

A Figura 5 mostra o mapa sendo importado para o ambiente RVIZ, onde o usuário pode determinar a posição de destino através da API (*Application Programming Interface*) chamada 2D Nav Goal.

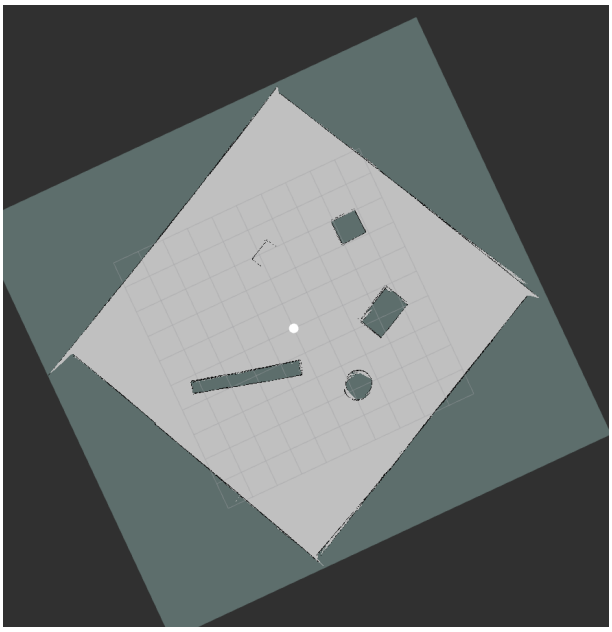

Fig. 5. Mapa importado para o ambiente RVIZ.

Fig. 6. Fluxograma do algoritmo do campo potencial.

B. Campo Potencial

O algoritmo possui seis etapas principais, conforme apresenta a Fig. 6:

Chamada da função (1): Toda vez que publicar no tópico *move_base_simple/goal* a função que gera os campos potenciais será chamada, ou seja, toda vez que inserir um novo destino

será executado o campo potencial.

Importação dos dados (2): Nessas etapa, importamos a posição de origem e destino junto com o mapa do ambiente.

Criação de outro mapa do tipo *float* (3): Nessa etapa, é criado um vetor do tipo de *float* com um tamanho reduzido, nesse mapa que será desenvolvido o campo potencial. É atribuído pesos para áreas livres (onde o robô pode trafegar) de 0.5, e os objetos recebem um peso de 1.0.

Inserção do destino (4): É gerado um círculo cujo centro deste círculo é o destino, isso será utilizado para aumentar a área do destino, esse artifício ajuda na hora da convergência do campo potencial. O destino recebe um peso de valor zero.

Suavização do mapa (5): É a etapa mais importante pois será nela que os gradientes do mapa são gerados. Sua função é atribuir valores a cada ponto do mapa em função da média dos pesos dos oito pixeis vizinhos, conforme mostra o algoritmo 1. Os pixeis que mudam de pesos, são especificamente os de áreas livres, o destino e os objetos mantêm seus pesos constantes. A suavização do mapa realizada apenas uma vez, porém foi atribuído quinze mil varreduras no processo de suavização. Após muitos testes foram definido este valor devido a quantidade de obstáculos presentes no mapa e o tamanho do mapa, este valor pode ser reduzido caso o mapa for menor ou a quantidade de objetos forem menores também. O algoritmo 1 mostra o desenvolvimento da suavização do campo potencial, nota-se que *smoothinglevel* é uma variável que assumiu um valor de 15000, ou seja, acontece 15 mil interações para gerar o campo. O destino e os objetos não participam das interações

Algorithm 1 Suavização do Campo Potencial

```
1: for k ← 0 at smoothinglevel do
2:     for i ← 1 at height-1 do
3:         for j ← 1 at width-1 do
4:             if (mapFloat[i*width + j] != Objetos)
   (mapFloat[i*width + j] != destino) then
   mapFloat[i*width + j] = float(mapFloat[(i+1)*width+
   j-1]+ mapFloat[(i)*width+ j-1]+mapFloat[(i-1)*width+
   j-1]+ mapFloat[(i+1)*width+ j]+mapFloat[(i-1)*width+
   j]+ mapFloat[(i+1)*width+ j+1]+ mapFloat[(i)*width+
   j+1]+mapFloat[(i-1)*width+ j+1])/8.0;
5:             end if
6:         end for
7:     end for
8: end for
```

Projeto do trajeto do robô (6): A partir da posição atual (cx e cy) importada pelo filtro de partícula, é analisada os 8 pixels vizinhos, onde é escolhido o pixel com peso menor, após é atualizada a posição atual para a posição do pixel escolhido. É realizado essa verificação de pesos até que a alcance a posição do destino como mostra o algoritmo 2.

C. Paralelização do Campo Potencial

Foi estudada uma forma de paralelizar o algoritmo de campos potenciais para aplicar em um MPSoC (*Multiprocessor Systems-on-Chip*), a Figura 7 exemplifica a paralelização, onde

Algorithm 2 Decisão da trajetória do robô

```
for i ← 1 at 8 do
    if (vector[i] ≤ vector[lowest_index]) thenlowest_index =
i;    end if
     if
         if  thenthen(mapFloat[cy*width + cx] == 0) break;
         else
             if (lowest_index==1) then cx–;
             end if
             if (lowest_index==0) then cy++; cx–;
             end if
             if (lowest_index==2) then cy–; cx–;
             end if
             if (lowest_index==3) then cy++;
             end if
             if (lowest_index==4) then cy–;
             end if
             if (lowest_index==5) then cy++; cx++;
             end if
             if (lowest_index==6) then cx++;
             end if
             if (lowest_index==7) then cy–; cx++;
             end if
             ROS_INFO("atual y : cy);
         end if
     end if
```

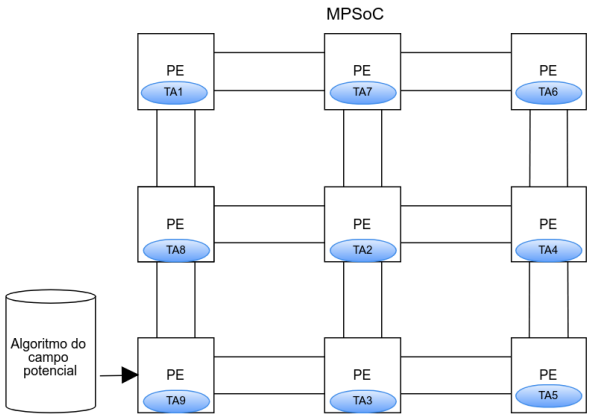

Fig. 7. Agoritmo do Campo potencial sendo executado num MPSoC.

o algoritmo do campo potencial é divido em tarefas (TA) que podem serem executadas independente umas das outras.

O objetivo principal era paralelizar a etapa de suavização, pois para que o campo potencial converja em menor tempo é necessaário um número alto de interações, como por exemplo, quinze mil interações em nossa implementação. Para isso, foi excluída a função *resize* do OpenCV e substituída por uma função (mostrada pelo Algoritmo 3)que reduz o mapa através de uma matriz 5x5, assim com essa nova função é possível paralelizar o comportamento. A função recebe uma matriz 5x5 do mapa original, representada pela variável *buffer*, o qual se prioriza o destino, ou seja, se na varredura da matriz encontra-se uma posição de destino, o valor retornado será o valor de destino, para outros valores como obstáculos e áreas livres é realizado um quórum.

Entretanto, ao analisar o paralelismo do algoritmo, notamos que para realizar a suavização é necessário o processamento sequencial dos pixels, o valor de um pixel influencia no outro, caracterizando assim, um processamento sequencial.

Algorithm 3 Função de redução do mapa

```
1: for i ← 0 at 5 do
2:     for j ← 1 at 5 do
3:         if buffer[i][j]≡preto) then return 0;
4:         else
5:             if buffer[i][j]==destino  then return 0;
6:             else
7:                 if buffer[i][j]==objetos then conta_branco++;
8:                 else
9:                     if buffer[i][j]==area_livre) then conta_area_livre++;
10:                    elsecontador_pixel++; somado_valor_cor=buffer[i][j];
11:                end if
12:            end if
13:        end if
14:
15:
16:        if conta_branco for maior then return objetos;
17:        else
18:            if  se area_livre for maior then return area_livre;
19:            end if
20:        end if
```

IV. Resultados

A Figura 7 mostra o trajetória do robøgerado pelo campo potencial, onde o círculo preto representa a posição de destino. Nota-se as curvas iniciais (em preto) do robô demonstram o alinhamento da frente do robô para o destino. importante ressaltar que os pontos pretos ao redor do destino são oriundos do processo de suavização. Observa-se que o robô saiu de sua origem e foi até o destino sem colidir com nenhum objeto.

V. Conclusão

Através dos resultados notamos que a técnica de campos potenciais demostrou uma grande acurácia na formação do trajeto proposto ao robô, podemos notar que a suavização do campo é uma etapa fundamental para que o robô encontre o destino em menor tempo.

É importante ressaltar, que durante a suavização, o desempenho foi melhor quando realizamos a média aritmética dos oitos pixels vizinhos ao invés de 4 pixels como fora implementado inicialmente. Fora analisado a paralelização desta técnica para executar em um MPSoC, porém há uma limitação ao criar o campo, pois é necessário o processamento sequencial para gerar o campo, onde o processamento de um pixel influência diretamente o pixel seguinte, limitando assim, a paralelização do algoritmo.

Agradecimentos

O autor agradece ao Pós-Doutorando Vitor A. M. Jorge e ao professor Alexandre Amory da Pontifícia Universidade Católica do Rio Grande do Sul pelas suas orientações para este trabalho.

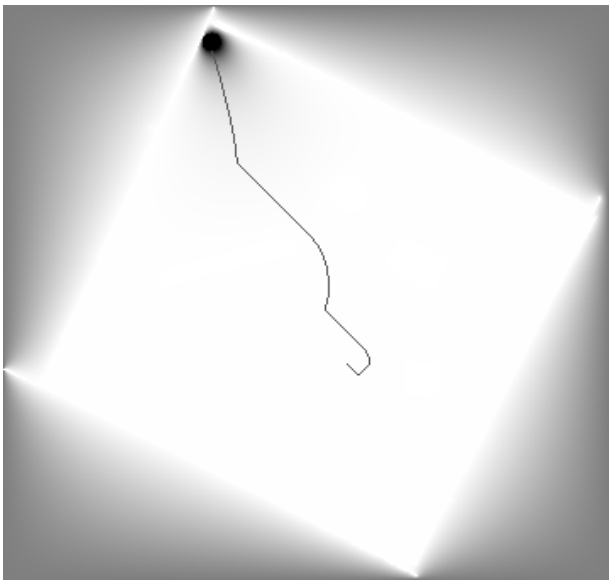

Fig. 8. O caminho a ser seguido pelo robô gerado pelos campos potenciais.

References

[1] Siegwart, R. Nourbakhsh, I. R. Introduction to Autonomous Mobile Robots, MIT Press, 2004.