CONEM 2010
VI Congresso Nacional de Engenharia Mecânica
ABCM COMPROMETIDA COM A PROMOÇÃO DO DESENVOLVIMENTO DA ENGENHARIA E DAS CIÊNCIAS MECÂNICAS
ASSOCIAÇÃO BRASILEIRA DE ENGENHARIA E CIÊNCIAS MECÂNICAS
VI CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA
VI NATIONAL CONGRESS OF MECHANICAL ENGINEERING
18 a 21 de agosto de 2010 – Campina Grande – Paraíba - Brasil
August 18 – 21, 2010 – Campina Grande – Paraíba – Brazil

O Uso de Algoritmos A* e Field A* em Robôs Agrícolas

Gabriel Queiroz Silva Abrahão, gabrielabrahao@yahoo.com.br
Poliane Torres Megda, politorresmegda@hotmail.com
Marcelo Becker, becker@sc.usp.br

Laboratório de Robótica Móvel - EESC USP, Av. Trabalhador são-carlense, 400, 13566-590, São Carlos – SP, Brasil.

Resumo: *Neste trabalho foram empregados e comparados os algoritmos A* e Field A* em aplicações de robótica móvel. O primeiro é um algoritmo simples e eficiente que permite que um robô móvel se mova até um destino predeterminado dados sua posição inicial, posição desejada e o mapa no qual o robô se encontra. O segundo, trata de uma extensão do algoritmo A* que leva a resultados mais precisos, porém com uma eficiência computacional menor (em termos de tempo de processamento). Em situações onde o mapa no qual o robô navega é conhecido e onde o ambiente não muda com o tempo (ambientes estáticos), como por exemplo, em uma plantação, os algoritmos A* e Field A* podem ser empregados para traçar a rota do robô agrícola exigindo um baixo custo computacional, quando comparados a algoritmos mais sofisticados que lidam com ambientes mais complexos e dinâmicos. O resultado do funcionamento de ambos os algoritmos é apresentado e o desempenho deles é comparado, em termos de rota gerada e comparação do tempo de processamento necessário para cada um deles calcular um caminho de uma posição inicial até um destino.*

Palavras chaves: *A*, Field A*, Navegação, Robótica Móvel, Robôs Agrícolas*

1. INTRODUÇÃO

Atualmente vários países e empresas do mundo investem uma grande quantidade de recursos na área da robótica móvel aplicada à agricultura. Isso ocorre principalmente devido à necessidade de aumentar a produção de alimentos, bem como reduzir os seus custos. Os robôs agrícolas podem ser utilizados para otimizar as atividades de colheita, de pulverização, de semeação e de preparo do solo. Porém, existem vários desafios que precisam ser superados para que seja produzido um robô agrícola economicamente viável. Um desses é a criação de um planejador de rotas que calcule uma trajetória suficientemente precisa para que o robô consiga realizar suas tarefas sem danificar a plantação, o que pode ocorrer quando por exemplo o robô caminha sobre ela. Um planejador de rota básico encontre uma série de estados que permitem um veículo sair de um ponto inicial e chegar a um destino. Um caminho otimizado consiste na sequência de transições de estado cuja soma dos custos de travessia é a menor possível. Na maioria dos casos, o planejamento de rotas em robótica móvel é complicado porque o ambiente no qual o robô tem de se locomover pode ser dinâmico. Para mapear o ambiente e detectar as mudanças que ocorrem no mesmo, é necessário que o robô possua um programa sofisticado, que exija um alto custo computacional e é difícil de ser desenvolvido. É necessário também que ele possua uma série de sensores embarcados, que são na maior parte dos casos muito caros. Dessa forma, a fabricação de robôs móveis para uso comercial acaba se restringindo a setores com capital suficiente para arcar com todos esses custos.

Entretanto, ao se considerar que o robô irá se mover em um ambiente estático e conhecido, a complexidade do problema é reduzida consideravelmente, permitindo o uso de um programa muito mais simples e requerendo uma quantidade menor de sensores. Isso pode ser feito em certas situações, como na agricultura de precisão. Em uma plantação, a área do plantio é bem delimitada e conhecida, assim como os espaços que serão usados para locomoção. Além disso, com raras exceções, o ambiente de uma plantação não muda com o tempo, podendo ser considerado estático. Algoritmos que permitem a obtenção de uma rota em um ambiente estático e conhecido ganham importância nesse contexto. Existem vários algoritmos que calculam uma trajetória de um ponto inicial até um destino em um ambiente estático e conhecido. Neste artigo, serão apresentados dois desses algoritmos, A* e Field A*. O algoritmo A* foi introduzido por Nilsson (1980), enquanto que o algoritmo Field A* é fruto de uma adaptação do algoritmo Field D*, o qual foi criado por Ferguson (2005). Exemplos de trajetórias fornecidas por ambos os algoritmos serão ilustrados e os desempenhos deles serão comparados em termos de rota gerada e do tempo requerido para a obtenção da mesma.

2. O ALGORITMO A*

Dado um mapa dividido em vários nós, classificados como transponíveis ou intransponíveis, e dado o ponto no qual um objeto se encontra e o ponto ao qual o objeto deseja ir, conforme a Fig. (1), o algoritmo A* calcula a sucessão de nós que compõem a rota de menor custo dentre todas as rotas compostas pelos nós.

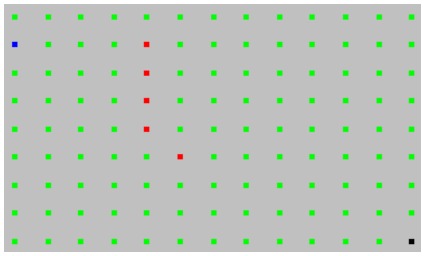

Figura 1 - Situação inicial, na qual em verde estão representados os nós livres, em vermelho os nós que não são transponíveis, em azul a origem e em preto o destino.

Para o cálculo dessa trajetória, o primeiro passo é a criação de uma lista aberta. Essa lista irá conter os nós que são candidatos a fazer parte do caminho final. Todos os nós presentes na lista aberta são organizados de acordo com um valor f, que representa a dificuldade prevista de sair da origem e chegar ao destino passando pelo nó, sendo que os nós com um valor menor de f são colocados no início da lista. Tal valor representado por f consiste na soma da dificuldade de se locomover da origem ao nó, que será chamada de g, somada à estimativa do custo de ir do nó ao destino, que será nomeada de h. Dessa forma, tem-se:

$$f = g + h \tag{1}$$

A estimativa h de sair de um nó qualquer e ir ao destino é calculada por meio de uma heurística. A heurística é uma função que estima o custo de ir de um nó até o ponto final a partir de parâmetros predeterminados. Isso ocorre porque o programa não tem meios de determinar a dificuldade real. Existem vários modos de prever esse custo, sendo que uma análise do contexto no qual o robô se encontra é necessária para escolher o método correto. Um exemplo é o método Manhattan, no qual a heurística é a soma da distância percorrida ao se locomover horizontalmente e depois verticalmente até o destino. Nesse método o movimento diagonal e possíveis obstáculos são desprezados.

Dado um nó A, existem diversas trajetórias que passam por A, as quais permitem um objeto sair da origem e ir ao destino. Dentre essas trajetórias, a que possui menor custo é chamada de trajetória otimizada. Existe um nó nessa trajetória que antecede o nó A. Esse nó recebe o nome de pai de A e o modo como ele é obtido será explicado mais adiante. O valor de g consiste no valor da distância do nó A ao seu pai, que em uma célula em forma quadrática com lado igual a um será um se eles não forem diagonais ou caso contrário pelo teorema de Pitágoras será raiz de dois, somado ao valor de g do pai de A. O procedimento anterior poderia ser repetido para calcular o g do pai de A, que por sua vez iria precisar do valor de g de outro nó, sendo que isso se repetiria até se chegar à origem, a qual não tem pai e cuja g é obviamente zero. Todavia, como será mostrado adiante, isso não é ncessário porque o valor de g do pai de qualquer nó já está salvo na memória do computador, tornando o programa mais eficiente em termos de tempo de processamento.

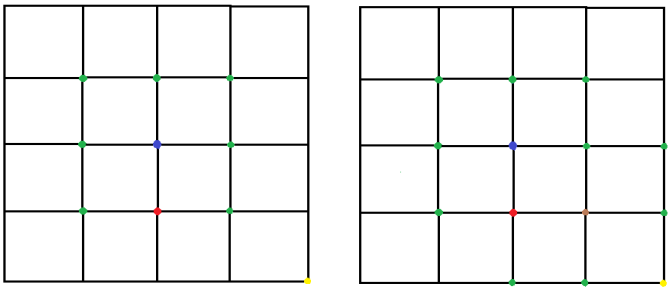

Figura 2. Funcionamento do algoritmo A*. Em verde estão os nós que foram colocados na lista aberta, em vermelho estão os nós intransponíveis, em azul a origem, em amarelo o destino e em marrom o nó que está sendo analisado.

Também é necessário criar uma lista fechada a qual irá conter os nós que já foram analisados pelo programa. A finalidade é evitar que esses nós, por terem normalmente um f pequeno, sejam analisados novamente, o que poderia gerar um loop infinito. O algoritmo A* funciona da seguinte forma: primeiro adiciona-se a origem à lista aberta anteriormente descrita. Feito isso, identifica-se todos os nós transponiveis adjacentes à origem, coloca-se na lista aberta, calcula-se o valor de f dos nós e organiza-se a lista aberta de acordo com f, conforme a Fig. (2) esquerda. Por fim, define-se a origem como o pai desses nós, ou seja, define-se que a origem pertencerá à trajetória de menor custo que sai da origem e passa por qualquer um desses nós. Em seguida adicione a origem à lista fechada e retire o primeiro nó A da lista aberta, que será o nó com o menor valor de f, conforme dito anteriormente. O motivo de esse nó ser selecionado é que quanto menor o valor de f, maior a chance de ele pertencer à rota de menor custo que leva ao destino. Uma vez feito isso, procura-se por nós transponíveis adjacentes ao nó A, que não pertencem à lista fechada, conforme a Fig. (2)

direita. Vê-se quais desses nós pertencem à lista aberta e quais não pertencem. Para os nós que não pertencem à lista, repete-se o procedimento feito com os nós adjacentes a origem, definindo-se o nó A como o pai deles. Para os outros nós, compara-se o valor atual de f desses com o valor obtido considerando o nó A como o pai. Se o valor atual for menor, mantém-se o pai. Caso contrário, define-se A como o pai daqueles e atualiza-se o valor de f dos mesmos.

O procedimento descrito no parágrafo anterior deve ser repetido até que a lista aberta não contenha nenhum nó ou que o destino seja adicionado à lista aberta. No primeiro caso, isso significa que não existe um caminho para a posição final. No segundo caso, a trajetória com o menor custo foi descoberta, sendo que ela é composta pelo destino, pelo nó A que é pai do destino, pelo nó B que é pai de A e assim por diante até se chegar à posição inicial.

Na Figura (3) é mostrado o resultado da execução de um programa que usa o algoritmo A*, sendo que o método usado na heurística é o método Manhattan, que foi descrito anteriormente.

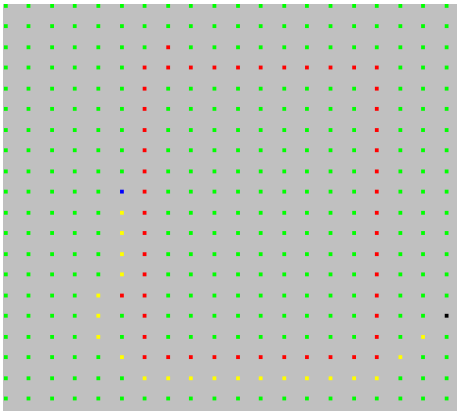

Figura 3. Rota calculada por um programa que implementa o algoritmo A*, na qual em verde estão representados os nós livres, em vermelho os nós que não são transponíveis, em azul a origem, em preto o destino e em amarelo a rota calculada.

3. O ALGORITMO FIELD A*

O algoritmo Field A* usa o mesmo princípio do algoritmo A*. A principal diferença entre esses algoritmos é que o segundo usa um mapa dividido em nós, enquanto o primeiro usa um mapa dividido em células. Cada lado das células será denominado de aresta. Sendo R a rota a qual possui o menor custo entre todas as rotas compostas pelos pontos contidos nas arestas do mapa, o algoritmo Field A* calcula a sucessão de arestas que contêm os pontos que formam uma rota de custo próximo ou igual ao custo da rota R. Note que, ao contrário do que ocorre no algoritmo A*, a rota A sempre será a trajetória de menor custo possível entre a origem e o destino.

No algoritmo Field A*, o pai de uma aresta A é definido como a aresta que precede a aresta A na sucessão de arestas que contêm A e que compõem o caminho otimizado entre a posição inicial e a posição final. Como ocorre no algoritmo A*, os conceitos de lista aberta e lista fechada e os valores f, g e h são usados no algoritmo Field A*. Porém, nesse algoritimo, quando é analisado um ponto de uma aresta, dois casos devem ser considerados. O primeiro caso ocorre quando o ponto B em análise se encontra em um nó. Nessa situação, a transição pode ocorrer para qualquer ponto que esteja localizado na borda de uma célula adjacente ao nó, conforme a Fig. (4).

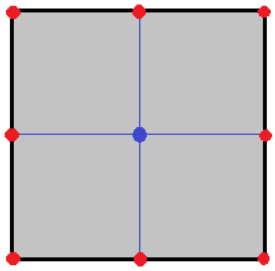

Figura 4. Células adjacentes a um nó central, representado em azul. As arestas nas quais o movimento é permitido estão representadas por uma linha preta.

Se fosse conhecido o valor da heurística para todos os pontos da borda dessa célula adjacente, o valor otimizado do ponto B poderia ser calculado minimizando o valor de c(B,C) + h(C), no qual c(B,C) representa o custo de sair do ponto B e ir a um ponto C qualquer localizado na borda de uma célula adjacente e h(C) consiste na heurística de C. Infelizmente existem infinitos pontos C, o que torna o cálculo de h(C) de cada um deles impossível. Dessa forma, o algoritmo Field A* usa interpolação linear para calcular o valor de h(C). Dado um ponto C residindo entre dois nós D e

E da aresta adjacente, a heurística de C é assumida como sendo a interpolação linear das heurísticas de D e E:

$$h(C) = y*h(E) + (1 - y)*h(D) \quad (2)$$

no qual y é a distância de D a C. Com essa aproximação pode-se determinar uma trajetória, a qual passa pelos pontos C e B, e que possui um custo próximo ao custo da trajetória otimizada.

O segundo caso ocorre quando o ponto B em análise se encontra no meio de uma aresta C. Todas as arestas contidas no mapa possuem duas ou pelo menos uma célula adjacente. Sendo X a célula adjacente à aresta C e que não possui a aresta pai de C,o movimento nesse caso limita-se aos pontos das arestas que são diferentes da aresta C e que estão contidas na célula X, conforme a Fig. (5). O motivo da transição não poder ser realizada para as células que contêm o pai de C é que não faz sentido o movimento acontecer em uma única célula mais de uma vez.

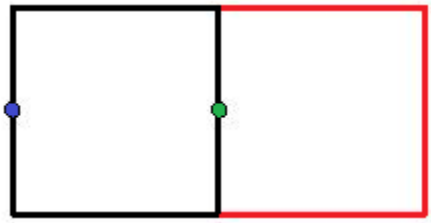

Figura 5. Células adjacentes a uma aresta. Em verde está o nó em análise, em azul o pai desse nó e em vermelho as arestas nas quais o movimento é permitido.

No segundo caso, dados o ponto B em questão e uma aresta D na qual o movimento é válido, traça-se uma reta E entre o ponto B e o destino. Se essa reta intersectar a aresta D, o ponto no qual essa intersecção ocorreu será o ponto que pertence à trajetória otimizada que passa por B e pela aresta D. Caso não intersecte, calcula-se a distância entre os nós da aresta D à reta E e determina-se qual dos nós está mais perto da mesma. Esse nó será o ponto que pertence ao caminho otimizado que passa pelo ponto B e pela aresta D. Na Figura (6) é mostrada uma iteração de um programa que implementa o algoritmo Field A*. Como o ponto está no meio da aresta, o método descrito no segundo caso foi utilizado para achar uma trajetória dentro da célula. Nota-se que o procedimento é semelhante ao usado no algoritmo A*. A Figura (7) ilustra o resultado do funcionamento de um programa que usa o algoritmo Field A*. Observa-se que a trajetória calculada pelo algoritmo não é a de menor custo, mas se aproxima dela.

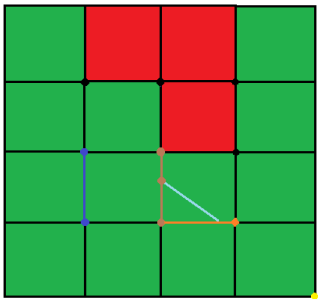

Figura 6. Ilustração de uma iteração do algoritmo Field A*, na qual em verde estão as células transponíveis, em vermelho as células intransponíveis, em amarelo o destino, em marrom a aresta A sendo analisada, em azul a aresta pai de A, em laranja uma aresta B adjacente à aresta A na qual o movimento é permitido e em cinza o caminho resultante da aresta A para a aresta B.

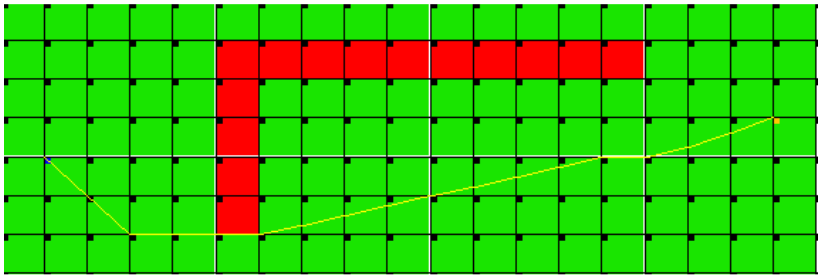

Figura 7. Caminho obtido por um programa que implementa o algoritmo Field A*, no qual em verde estão as células transponíveis e em vermelho as células intransponíveis. O nó azul representa a origem e o nó amarelo representa o destino.

4. RESULTADOS

Usando os métodos descritos, os algoritmos A* e Field A* foram implementados em uma mesma máquina nas mesmas condições. A Tabela (1) mostra os resultados obtidos em termos de tempo e distância em três casos distintos. Esses casos estão ilustrados nas Figuras (8), (9) e (10). Nota-se que não há unidade métrica na distância, já que esta unidade seria igual à distância entre dois nós adjacentes não diagonais do mapa fornecido ao algoritmo.

Tabela 1. Resultados das simulações dos algoritmos A* e Field A* em três casos distintos.

	Tempo A* (ms)	Tempo Field A*(ms)	Distância A*	Distância Field A*
Caso 1	**5**	**5**	**22,360**	**20,126**
Caso 2	**5**	**10**	**36,828**	**35,928**
Caso 3	**5**	**6**	**12,485**	**11,656**

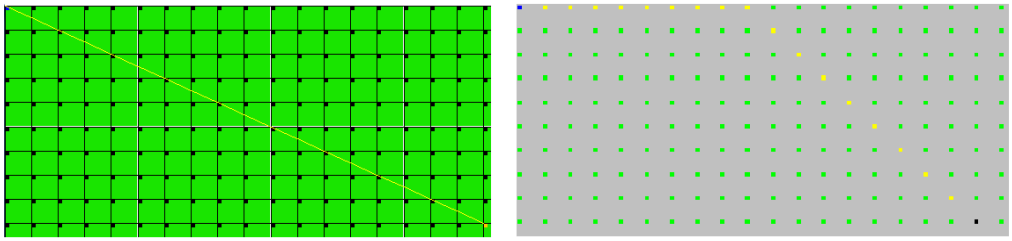

Figura 8. Resultados dos algoritmos A*(direita) e Field A*(esquerda) na obtenção da rota em um retângulo de comprimento igual a 18 unidades e largura igual a 9 unidades.

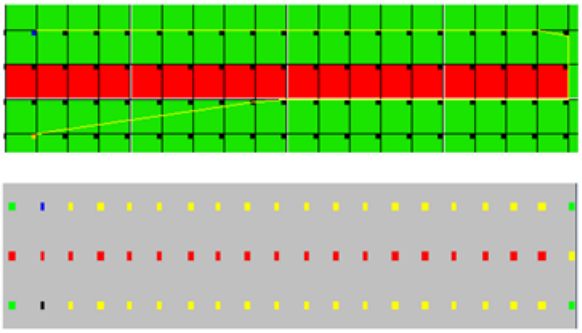

Figura 9. Resultados dos algoritmos A*(em baixo) e Field A*(em cima) na obtenção de uma trajetória na transposição de uma coluna de comprimento igual a 18 unidades.

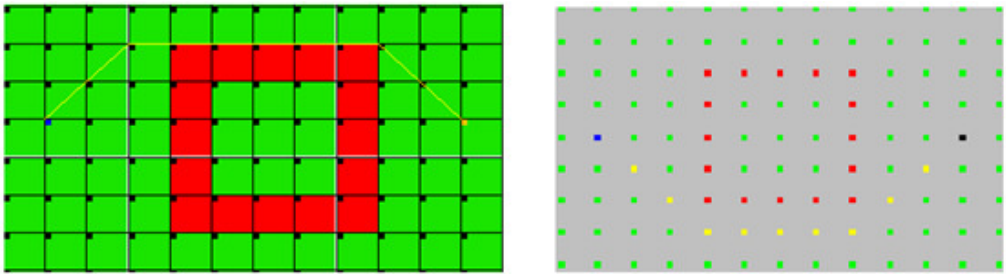

Figura 10. Resultados dos algoritmos A*(direita) e Field A*(esquerda) na obtenção de uma trajetória na transposição de um quadrado de lado igual a 5 unidades.

Os resultados mostram, conforme esperado, que o algoritmo Field A* exige uma capacidade de processamento maior, porém alcança resultados mais precisos. É importante ressaltar, entretanto, que os testes não foram muito precisos devido à diferença de posicionamento da origem e do destino nos dois algoritmos. No algoritmo A*, os nós foram colocados no centro de uma célula, enquanto que no algoritmo Field A* o posicionamento foi feito nos nós das arestas. Dessa forma, sendo B a trajetória de menor custo dentre todas as trajetórias possíveis entre a origem e o destino, temos que no segundo teste a rota B do algoritmo Field A* ficou com um tamanho superior à rota B do algoritmo A*, enquanto que no terceiro caso ocorreu o inverso.

Poderiam ser obtidos resultados mais precisos se o programa que implementa o algoritmo Field A* fosse melhorado de forma que a origem e o destino pudessem ser colocados em qualquer lugar do mapa. Isso, alias, é um problema do algoritmo A*. Nesse algoritmo, a origem e o destino devem ser colocados em um nó, o que pode gerar problemas de localização e uma perda de precisão. Por fim, destaca-se que o programa o qual implementou o algoritmo Field A* provavelmente pode ser melhorado, de modo que o mesmo possa obter trajetórias mais precisas ou requerer uma capacidade de processamento menor.

5. AGRADECIMENTOS

Os autores gostariam de agradecer ao CNPq e à FINEP pelo apoio financeiro durante a execução desse trabalho.

6. REFERÊNCIAS

Ferguson, D and Stentz, A, 2005, “Field D*: An Interpolation-based Path Planner and Replanner”, Proceedings of the International Symposium on Robotics Research (ISRR).

Nilsson, N, 1980, "Principles of Artificial Intelligence", Tioga Publishing Company, pp. 72-88.

The Use of Algorithms A* and Field A* in Agricultural Robots

Gabriel Queiroz Silva Abrahão, gabrielabrahao@yahoo.com.br
Marcelo Becker, becker@sc.usp.br

Laboratório de Robótica Móvel - EESC USP, Av. Trabalhador são-carlense, 400, 13566-590, São Carlos – SP, Brazil.

Resumo: *In this work the algorithms A* and Field A* were implemented and compared in mobile robotic applications. The first is a simple and efficient algorithm that allows a mobile robot to move to a determinate destine given his initial position, the desire position and the map which the robot is moving. The second is an extension of the algorithm A* which leads to more precise results, however with a lower computational efficiency (in terms of processing time). When the map which the robot is moving is known and the environment don't change with time (static environment), like for example in a plantation, the algorithms A* and Field A* can be employed to find a route of a mobile robot demanding a low computational cost when compare to more sophisticate algorithms which works with more complex and dynamic environments. The result of the operation of both algorithms is present and their performance is compare, in terms of generate route and processing time required for each of them find a way between a start point and a goal position.*

Keywords: *A*, Field A*, Navegation, Mobile Robotics, Agricultural Robotics*