X WORCAP
20 e 21 de outubro
Instituto Nacional de Pesquisas Espaciais - INPE
São José dos Campos - SP

Aplicação de Algoritmo Dijkstra ao Planejamento de Movimento de VANTs

Felipe Leonardo Lôbo Medeiros[12], José Demisio Simões da Silva[2]

[1]Instituto de Estudos Avançados (IEAv)
Caixa Postal 6044 – 12.228-970 – São José dos Campos – SP – Brasil

[2]Instituto Nacional de Pesquisas Espaciais (INPE)
Caixa Postal 515 – 12227-010 – São José dos Campos – SP – Brasil
felipe@ieav.cta.br, demisio@lac.inpe.br

Abstract. *This work presents results of the application of a Dijkstra algorithm to the planning of trajectories for fixed-wing Unmanned Aerial Vehicles (UAVs). The navigation environments are represented by sets of visibility graphs constructed through the terrain elevations of these environments. Digital elevation models are used to represent the terrain elevations. The trajectories planned by this algorithm are collision-free and dynamically feasible.*

Resumo. *Este trabalho apresenta resultados da aplicação de um algoritmo Dijkstra ao planejamento de trajetórias para Veículos Aéreos Não Tripulados (VANTs) de asa fixa. Os ambientes de navegação são representados por conjuntos de grafos de visibilidade construídos através de elevações de terreno destes ambientes. Modelos digitais de elevação são utilizados para representar as elevações de terreno. As trajetórias planejadas por este algoritmo são livres de colisão e dinamicamente viáveis.*

Palavras-chave: *VANT de Asa Fixa, Planejamento de Movimento, Algoritmo Dijkstra, Modelo Digital de Elevação, Busca Local Baseada em Grade.*

1. Introdução

O planejamento automático de movimento ou trajetória é essencial para diversas tarefas que conduzem ao aumento de autonomia de Veículos Aéreos Não Tripulados (VANTs). Este trabalho aborda o problema de planejamento de trajetórias para VANTs de asa fixa considerando constantes a altitude de navegação e o raio de curva destes veículos. Este problema é similar ao planejamento de trajetórias para robôs móveis não holonômicos que possuem rodas [Anderson *et al*. 2005]. Uma trajetória é uma seqüência de coordenadas de navegação (*waypoints*) conectadas por segmentos de reta e arcos e que

permite ao VANT de asa fixa navegar entre um *waypoint* de origem e um *waypoint* de destino do ambiente de navegação. Uma trajetória deve ser livre de colisão e dinamicamente viável [Anderson *et al*. 2005], isto é, uma trajetória deve permitir ao VANT navegar entre dois *waypoints* sem violar as restrições cinemáticas do veículo. Uma definição formal de trajetória dinamicamente viável é apresentada em [Anderson *et al*. 2005]. A Fig. 1 apresenta um exemplo de trajetória livre de colisão e dinamicamente viável planejada para um VANT de asa fixa, considerando constantes o raio de curva e a altitude de navegação. Os polígonos pretos representam as regiões não navegáveis do ambiente de navegação. Os círculos vermelhos representam os *waypoints* da trajetória. Os círculos cinza tracejados indicam os círculos definidos pelo raio de curva do VANT.

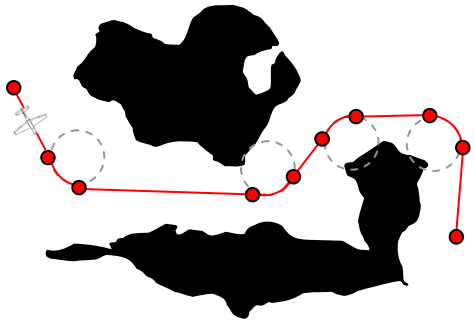


Figura 1. Exemplo de uma trajetória planejada para um VANT de asa fixa. Fonte: [Medeiros e Silva 2010]

Uma revisão recente de métodos para o planejamento de trajetórias para VANts é apresentado em [Goerzen *et al*. 2010]. Como mencionado nesta revisão, o algoritmo Dijkstra [Dijkstra 1959] garante solução ótima quando usado no planejamento de rotas. Uma rota é uma seqüência de *waypoints* conectados por segmentos de reta e que é utilizada para definir uma trajetória para um veículo. As rotas planejadas pelo algoritmo Dijkstra não consideram as restrições cinemáticas de um VANT e devem ser transformadas em trajetórias através da aplicação de métodos de suavização.

Um algoritmo Dijkstra baseado em Elevações (ADE) de terreno foi proposto em [Medeiros e Silva 2010] visando o planejamento de trajetórias dinamicamente viáveis e livres de colisão, através de grafos de visibilidade e modelos digitais de elevação. O ADE é uma adaptação do Algoritmo Dijkstra Modificado (ADM) proposto em [Kwata e How 2004] e apresenta três principais diferenças em relação a este algoritmo. A primeira diferença é que o ADE tem como objetivo a solução do problema de trajetória mais curta entre dois pontos. A segunda diferença é que o ADE planeja trajetórias da classe *k*-trajetórias, proposta em [Anderson *et al*. 2005]. A última diferença é que o ADE possui uma heurística de busca local e baseada em grade para verificar se um arco de uma trajetória é livre de situação de colisão. Esta heurística é invariante ao número de obstáculos e apresenta tempo computacional $O(n_p)$, no pior caso, onde n_p é o número de iterações necessárias para uma verificação. Considerando a utilização de busca binária para a determinação do nó de menor custo, o ADE apresenta tempo computacional igual a $O(n_p n_e log(n_n))$, no pior caso, onde n_e e n_n são o número de arestas e de nós de um grafo de visibilidade.

A seguir é apresenta a organização do restante deste artigo. A Seção 2 aborda a construção de grafos de visibilidade que representam possíveis rotas de ambientes de

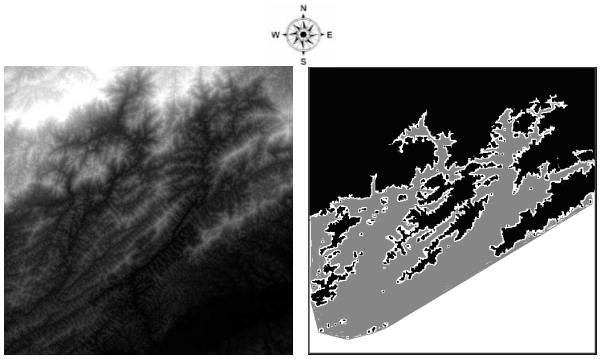
navegação. A Seção 3 apresenta o ADE e os resultados da aplicação deste algoritmo. Conclusões finais baseadas nos resultados obtidos são apresentadas na Seção 4.

2. Grafos de Visibilidade

Em problemas de planejamento de rotas, um grafo de visibilidade é um conjunto de nós ou *waypoints* definidos pelos vértices convexos dos obstáculos do ambiente de navegação e conectados por arestas que não interceptam tais obstáculos [Nilsson 1969].

Este trabalho utiliza a mesma metodologia apresentada em [Medeiros e Silva 2010] para a construção de grafos de visibilidade. Um conjunto de possíveis rotas livres de colisão de um único ambiente de navegação é representado por um conjunto de grafos de visibilidade. Os nós e arestas dos grafos de visibilidade são determinados pela aplicação de dois algoritmos propostos em [Medeiros e Silva 2008] e que constroem conjuntos de grafos de visibilidade diretamente de modelos digitais de elevação, considerando constantes uma altitude a_n, uma altura a_s, e uma distância de segurança d_s entre um nó e a extremidade que define este nó. Os nós são coordenadas geográficas e as regiões não navegáveis são definidas pelas células e_{lc} com elevação superior ou igual a $a_n - a_s$, onde a_n é a altitude de navegação, e a_s é a altura de segurança.

Estes algoritmos foram aplicados à construção de grafos de visibilidade baseados no modelo digital de elevação da Fig. 2a. A matriz de elevações E deste modelo é matriz quadrada de ordem 1205 e possui resolução r_s = 30 metros (m), isto é, cada célula e_{lc} representa uma região com altura e largura igual a 30 m. Um resultado da aplicação destes algoritmos é apresentado na Fig. 2b, considerando a_n = 1100 m, a_s = 300 m e d_s = 100 m. Os polígonos pretos indicam as regiões não navegáveis. Os círculos e segmentos de reta cinza representam, respectivamente, os nós e arestas dos grafos de visibilidade. O conjunto de grafos de visibilidade da Fig. 2b é composto por n_n = 1489 nós e n_e = 65454 arestas, e o ambiente de navegação apresenta n_o = células não navegáveis.



(a) (b)

Figura 2. (a) Modelo digital de elevação usado neste trabalho. (b) Conjunto de grafos de visibilidade construído. Adaptado de: [Medeiros e Silva 2010]

3. Algoritmo Dijkstra Baseado em Elevação de Terreno

O ADE planeja *k*-trajetórias para VANTs de asa fixa com base em grafos de visibilidade e modelos digitais de elevação do ambiente de navegação. Uma *k*-trajetória é uma curva de Dubins [Dubins 1957] cujo comprimento e forma podem ser ajustados por um parâmetro *k*. Dada uma rota composta pelos *waypoints* w_{i-1}, w_i e w_{i+1}, uma *k*-trajetória é a única trajetória de menor extensão, dinamicamente viável, e que permite a navegação

do *waypoint* w_i para o *waypoint* w_{i+1}, passando diretamente por *p(k)* [Anderson *et al*. 2005]. Um exemplo de *k*-trajetória é apresentada na Fig. 3a. As trajetórias são representadas pelos segmentos de reta, arcos e círculos vermelhos.

A trajetória mais curta dentre as *k*-trajetórias é aquela em que $k = 1$. Devido a este fato, o ADE planeja *k*-trajetórias, para $k = 1$ (1-trajetória), como apresentado na Fig. 3b. Entretanto, o ADE pode ser adaptado simplesmente para outros valores de *k*. O ADE é apresentado na Tabela 1, onde: *d'* é a distância entre o nó de origem ou nó inicial w_s e o nó analisado; p_r é o nó anterior ao nó analisado; r_c é o raio de curva do VANT; e h_{cs} é a heurística para verificar se um arco do *waypoint* a_e para o *waypoint* a_s é livre de colisão.

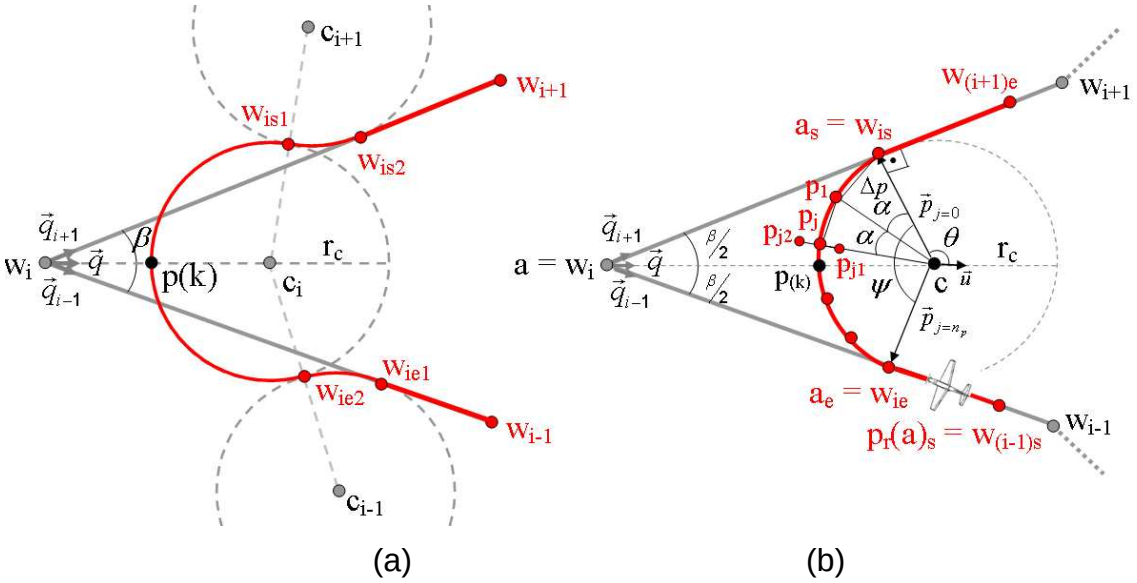


Figura 3. (a) Exemplo de uma *k*-trajetória de w_{i-1} para w_{i+1}. (b) Discretização do arco que conecta w_{ie} a w_{is} de uma 1-trajetória planejada pelo ADE. Fonte: [Medeiros e Silva 2010]

Desta forma, analisando a Fig. 3b, uma 1-trajetória planejada pelo ADE através de um grafo de visibilidade é composta por: um segmento de reta $\overline{w_1 w_{2e}}$; um arco de w_{2e} para w_{2s}; uma seqüência de segmentos de reta $\overline{w_{(i-1)s} w_{ie}}$ conectados por arcos de w_{ie} para w_{is}; e um segmento de reta $\overline{w_{(n_t-1)s} w_{n_t}}$, para *i* variando de 3 a n_t - 1, onde n_t é o número de elementos da trajetória planejada. Na Fig. 3, $\vec{q}$ é um vetor unitário definido pelos vetores unitários $\vec{q}_{i-1}$ e $\vec{q}_{i+1}$. Os *waypoints* w_1 e w_{n_t} são a origem w_s e o destino w_d, respectivamente. Os *waypoints* w_l são nós do grafo de visibilidade, para *l* variando de 1 até n_n. Os *waypoints* w_{ie} e w_{is} são, respectivamente, o *waypoint* de entrada e o *waypoint* de saída da curva definida pelo raio de curva do VANT.

As restrições descritas na linha 10 do ADE garantem que uma 1-trajetória planejada pelo algoritmo seja dinamicamente viável. Cada $\overline{w_{(i-1)s} w_{ie}}$ é um segmento de reta da aresta que conecta os nós w_{i-1} e w_i do grafo de visibilidade. Este fato assegura que os segmentos de reta de uma trajetória planejada pelo ADE sejam sempre livres de colisão. Entretanto, o grafo de visibilidade não assegura que os arcos da trajetória planejada sejam livres de colisão. Portanto, a linha 13 do ADE é aplicação de uma

heurística para verificar se o arco entre dois *waypoints* gera uma situação de colisão com regiões não navegáveis. Esta heurística é baseada em espaço de estados mencionada em [Sanchez e Latombe 2003].

Tabela 1. Algoritmo Dijkstra baseado em elevação. Fonte: [Medeiros e Silva 2010]

Índice	Algoritmo Dijkstra Baseado em Elevação (ADE)
1	para i variando de 1 até n_n faça
2	armazenar w_i em Q
3	$d'(w_i) \leftarrow \begin{cases} 0, para\ w_i = w_s \\ \infty, para\ w_i \neq w_s \end{cases}$
4	$a \leftarrow w_s$
5	enquanto $a \neq w_d$ faça
6	remover a de Q
7	para cada vizinho v de a faça
8	$f \leftarrow d'(a) + d(a,v)$
9	determinar a_e, a_s, $p_r(a)_s$ através de $p_r(p_r(a))$, $p_r(a)$, a e v
10	se $d(a,a_s) > d(a,v_e)$ ou $d(a,a_e) > d(a,p_r(a)_s)$ faça
11	$f \leftarrow \infty$
12	senão
13	se $h_{cs}(a_e, a_s, E) = 1$ faça
14	$f \leftarrow \infty$
15	se $f < d'(v)$ faça
16	$d'(v) \leftarrow f$
17	$p_r(v) \leftarrow a$
18	$a \leftarrow min\ (d'(w_i))$, para $w_i \in Q$
19	enquanto $a \neq w_s$ faça
20	armazenar a na pilha T
21	$a \leftarrow p_r(a)$

Esta heurística verifica se um VANT de asa fixa navegando através de um arco intercepta alguma célula não navegável, isto é, uma célula $e_{lc} \geq (a_n - a_s)$, através de uma busca local em E. Como apresentado na Fig. 3b, a heurística discretiza o arco de a_e até a_s em uma seqüência de *waypoints* p_j, para j variando de 1 até $n_p = \psi/\alpha$. Analisando a Fig. 4, a cada iteração j, a heurística encapsula o movimento entre dois *waypoints* p_{j-1} e p_j em um pentágono, e verifica se este pentágono intercepta alguma célula não navegável. Nesta figura, os arcos pretos representam a trajetória do VANT e o pentágono é representado pelos segmentos de reta cinza tracejados.

Como apresentado e provado em [Medeiros e Silva 2010], o comprimento ζ e a envergadura ξ do VANT, a distância entre os *waypoints* p_{j-1} e p_j, e os comprimentos dos lados do pentágono devem ser inferiores à resolução r_s, com a finalidade de garantir a verificação de situações de colisão a cada iteração j. A Eq. 1 assegura esta restrição de distância.

$$\alpha = 2\,arcsin\left(\frac{\Delta p}{2r_c}\right) \tag{1}$$

A Seção 3.1 apresenta os resultados da aplicação do ADE.

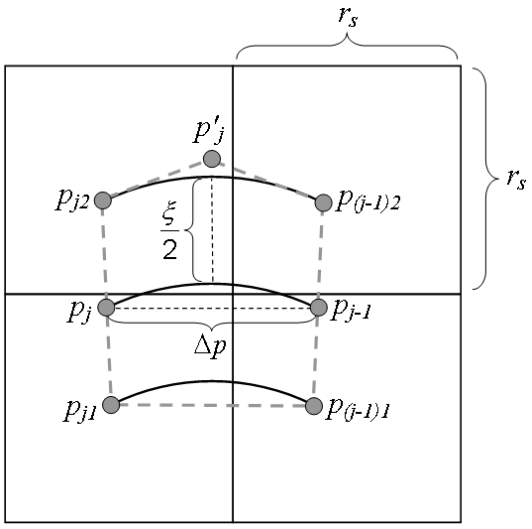


Figura 4. Verificação de colisão através de um pentágono que contém a trajetória do VANT entre os *waypoints* p_{j-1} e p_j. Fonte: [Medeiros e Silva 2010]

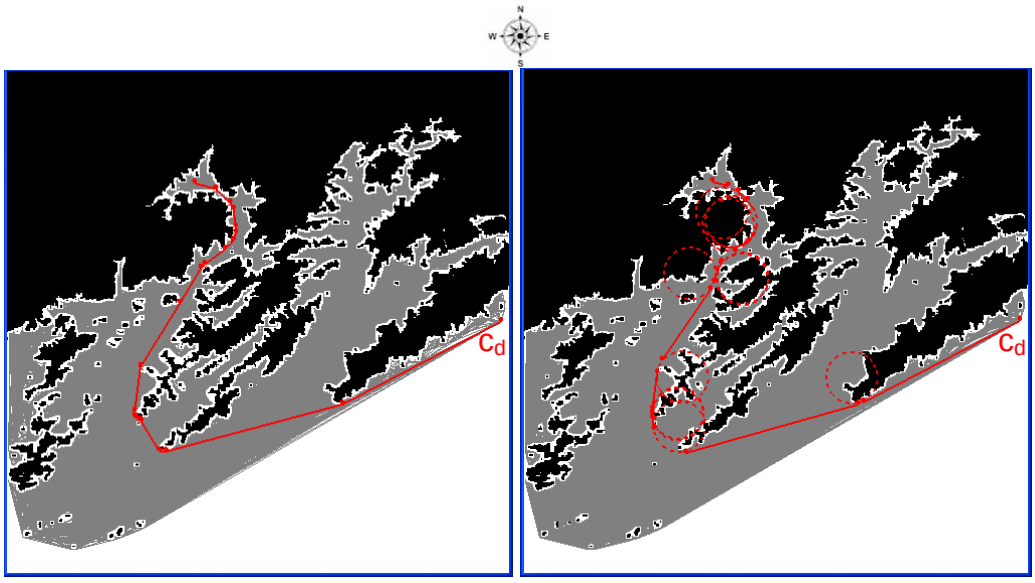
3.2 Aplicação do ADE

ADE foi aplicado ao planejamento de trajetórias considerando os conjuntos de grafos de visibilidade obtidos na Seção 2.

Inicialmente, um conjunto de especificações foi considerado para a aplicação do ADE. Analisando a Eq. 1, pode-se observar que n_p diminui quando o valor de Δp aumenta. Devido a este fato, o valor de Δp é especificado como $0.99r_s$ na aplicação do algoritmo. Analisando a Fig. 3b, observa-se que $0^o < \psi < 180^o$. Então, considerou-se que o pior caso para n_p ocorre quando: ψ é igual a 179.99^o; e a cada iteração j da heurística, três células são avaliadas para cada segmento de reta do pentágono. Portanto, no pior caso, o número de células verificadas pela heurística é $n_c = 7n_p = 7(179.99/2\,arcsin(0.99r_s/2r_c))$, considerando que o segmento de reta $\overline{p_{(j-1)1}p_{(j-1)2}}$ e alguns é avaliado na iteração anterior.

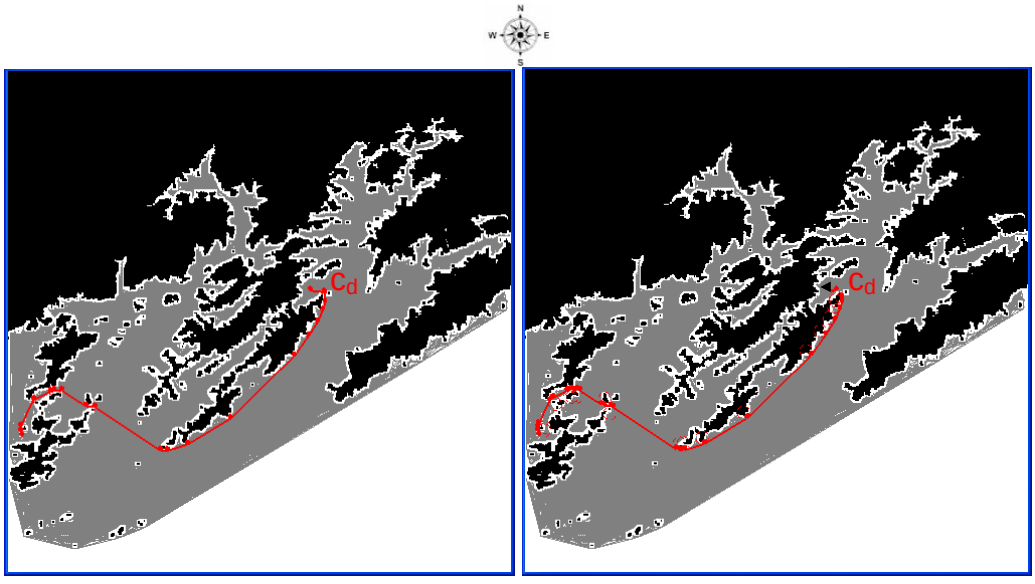
No pior caso, a heurística do ADE para verificação de colisão em um arco executa $n_p = (179.99/2\,arcsin(0.99(r_s = 30)/2(r_c = 300))) = 32$ iterações e explora $n_c = 7(n_p = 32) = 224$ células, considerando o conjunto de grafos de visibilidade da Fig. 2b.

Exemplos de 1-trajetórias planejadas pelo ADE são apresentados na Fig. 5 e na Fig. 6. As trajetórias são representadas pelos segmentos de reta, círculos e arcos tracejados vermelhos. Os arcos das trajetórias planejadas são seções dos círculos vermelhos tracejados definidos pelo raio de curva r_c. Como demonstrado na seção anterior, estas 1-trajetórias são livres de colisão e dinamicamente viáveis para VANTs de asa fixa com as seguintes restrições: $\xi < r_s = 30$ m; $\zeta < 30$ m; e $r_{cmin} \leq r_c \leq r_{cmax}$, onde r_{cmin} e r_{cmax} são o raio de curva mínimo e máximo, respectivamente.



(a) (b)

Figura 5. 1-trajetórias planejadas pelo ADE, considerando: d_s = 100 m; w_s = (latitude, longitude) = (-22.91, -45.95); w_d = (-23.0, -45.735); (a) r_c = 400 m; e (b) r_c = 2000 m



(a) (b)

Figura 6. 1-trajetórias planejadas pelo ADE, considerando: d_s = 100 m; w_s = (-23.075, -46.072); w_d = (-22.98, -45.87); (a) r_c = 300 m; e (b) r_c = 500 m

Em todos os testes, o tempo necessário para o planejamento de cada trajetória foi inferior a 2.95 segundos (s). Os testes foram efetuados em um computador com um processador de 1.7 GHz e com 1 GB de memória RAM.

4. Conclusões

Os resultados obtidos comprovam a eficiência do ADE para planejar 1-trajetórias para VANTs de asa fixa. Estas trajetórias são dinamicamente viáveis e livres de situações de colisão com obstáculos do ambiente de navegação.

O algoritmo pode ser aplicado considerando outros tipos de grafos usados como forma de representação do ambiente de navegação. A única restrição é que as arestas destes grafos sejam livres de colisão.

O algoritmo possui uma heurística de busca local para verificar situações de colisão que é baseada no modelo digital de elevações do ambiente de navegação. Esta heurística pode ser adaptada para grades não regulares, desde que Δp seja menor que o comprimento do menor lado dentre todas as células da grade.

Referências

Anderson, E. P., Beard, R. W., McLain, T. W. (2005) “Real-Time Dynamic Trajectory Smoothing for Unmanned Air Vehicles”, IEEE Transactions on Control Systems Technology, vol. 13, no. 3, p. 471-477.

Goerzen, C., Kong, Z., Mettler, B. (2010) “A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance”, In: Journal of Intelligent and Robotic Systems, p. 65–100. Springer, The Netherlands.

Kuwata, Y., How, J. P. (2004) “Stable Trajectory Design for Highly Constrained Environments Using Receding Horizon Control”, Proceeding of the 2004 American Control Conference, Boston, Massachusetts, p. 902-907.

Dijkstra, E. W. (1959) “A Note on Two Problems in Connection with Graphs”, In: Numerische Mathematik, 1, p. 269–271.

Dubins, L. E. (1957) “On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents”, American Journal of Mathematics, vol. 79, no. 3, p. 497-516.

Medeiros, F. L. L., Silva, J. D. S. (2008) “Grafos de Visibilidade Aplicados à Representação Computacional de Ambientes de Navegação Aérea”, In: X– Simpósio de Aplicações Operacionais em Áreas de Defesa (SIGE), São José dos Campos – SP.

Medeiros, F. L. L., Silva, J. D. S. (2010) “A Dijkstra Algorithm for Fixed-Wing UAV Motion Planning Based on Terrain Elevation”, In: 20^{o} Simpósio Brasileiro de Inteligência Artificial (SBIA), São Bernardo do Campo,SP, Brasil. Springer Lecture Notes in Computer Science. aceito para publicação.

Nilsson, N. J. (1969) "A Mobile Automaton: An Application of Artificial Intelligence Techniques”. In Proceedings of the International Joint Conference on Artificial Intelligence , Association for Computing Machinery, New York, p. 509-520.

Sanchez, G., Latombe, J. C. (2003) “A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking”, Published in Robotics Research: The Tenth Int. Symp., R.A. Jarvis and A. Zelinsky (eds.), Springer Tracts in Advanced Robotics, Springer, p. 403-417.