ANÁLISE DE ALGORITMOS DE PATH-PLANNING

Leonardo da Silva Costa[1], Flávio Tonidandel [2]
[1] Engenharia de Automação e Controle, Centro Universitário FEI
[2] Ciência da Computação, Centro Universitário FEI
unieleocosta, flaviot {@fei.edu.br}

Resumo: O projeto propõe a implementação e análise dos algoritmos A Estrela (A*) e *Rapidly-exploring Random Trees* (RRT) a fim de decidir qual o melhor para ser aplicado no futebol de robôs da categoria *RoboCup – Small Size League* (SSL). A partir dos resultados encontrados foi possível comprovar que o A* superou o RRT nos quesitos de velocidade de cálculo e comprimento do trajeto gerado, que são de grande relevância para a liga SSL.

1. Introdução

A categoria *Small Size League* (SSL) é uma das mais dinâmicas da *RoboCup* devido à alta velocidade da bola, que pode chegar à 6,5 m/s, e dos robôs, que chegam à aproximadamente 2 m/s. Para que esta velocidade de jogo seja possível é preciso que sejam utilizados algoritmos de cálculo de trajetória (*path-planners*) a fim de evitar colisões entre os robôs e em alguns casos com a bola também. É importante que os robôs colidam o menor número de vezes possível, pois de acordo com as regras [3] uma colisão cuja diferença de velocidade entre os robôs seja maior que 1,5 m/s será atribuída uma falta ao robô mais rápido e isso resulta em uma cobrança de chute em favor do oponente.

A fim de decidir qual seria a melhor escolha para a equipe RoboFEI-SSL foram estudados os algoritmos A* e RRT por serem utilizados em outras equipes da categoria, e pela simplicidade na implementação.

2. A Estrela

O A Estrela (A*) [1] é um algoritmo do tipo determinístico que faz o cálculo dos trajetos utilizando uma função de avaliação como na Equação 1.

$$f(v) = h(v) + g(v) \quad (1)$$

Onde, $h(v)$ é o valor da heurística calculada do estado atual v até o destino, $g(v)$ é o custo para se mover do estado anterior até o estado atual.

Para encontrar um caminho é feito o cálculo de $f(v)$ para 8 estados vizinhos do estado atual. Os estados que estão livres são colocados em uma lista denominada *open list*, o estado seguinte será o que estiver na *open list* e possuir o menor valor de $f(v)$ e por fim esse estado será colocado em outra lista chamada de *closed list* e a partir dessa lista será extraído o trajeto final quando o algoritmo chegar ao destino.

O A* foi escolhido pois, embora ele tenha sido inventando há cerca de 50 anos ele ainda é capaz de produzir bons resultados em certas situações, como é o caso do SSL. Além disso, seu pseudo-algoritmo é de fácil compreensão e implementação. Um exemplo do caminho feito pelo algoritmo pode ser visto na Figura 1 em vermelho.

3. Rapidly-exploring Random Trees

O *Rapidly-exploring Random Trees* (RRT) [2] é um algoritmo do tipo estocástico ou probabilístico, isso significa que o mesmo encontra os trajetos de forma aleatória e nem sempre é capaz de encontrar um trajeto válido. O algoritmo é composto dos seguintes passos: é gerado um ponto aleatório no espaço de busca, na sequência busca-se qual o vértice da árvore está mais perto deste ponto e então expande-se este vértice em direção ao ponto aleatório utilizando uma distância padrão pré-definida. Como não são feitos cálculos para saber onde está o destino esse algoritmo nem sempre chegará exatamente no destino, e sim em um ponto próximo, essa distância do destino é pré-definida pelo usuário, quanto menor ela for mais tempo levará para encontrar um caminho. Na Figura 1 é possível ver o resultado gerado pelo algoritmo, em verde têm-se a árvore gerado, o caminho sem tratamento em cinza e o caminho final em preto.

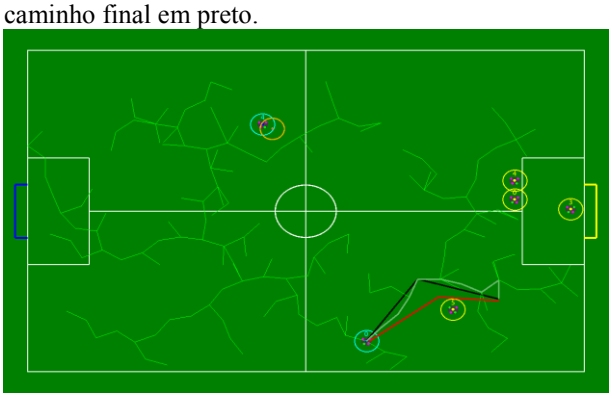

Figura 1- Resultado final RRT e A*.

O RRT foi escolhido por ser bastante simples, assim como o A*, e também por ser utilizado em diversas equipes da categoria SSL como a *CMDragons*, que utiliza uma variação do RRT denominada *Extended RRT* (ERRT) [4].

4. Metodologia

Os algoritmos foram implementados na linguagem C++ utilizando a IDE *QtCreator* no ambiente Linux.

Foram realizados testes no simulador GrSim [5] e no campo real para medir o desempenho de cada um dos algoritmos em relação ao tempo de cálculo e comprimento do trajeto. O desempenho foi medido baseado no tempo de cálculo, adquirido por meio de recursos disponíveis na IDE *QtCreator*, e também do comprimento do trajeto gerado, calculado a partir do caminho final gerado pelos algoritmos após a redução de pontos desnecessários.

5. Resultados

Foram testadas três situações, sendo elas, interceptar um passe, mover-se de forma a receber um passe e retornar para a região de defesa. Para cada uma das situações testadas foram coletadas cerca de 10.000 amostras, na Tabela I têm-se os resultados para o A* e na Tabela II para o RRT em cada uma das três situações.

Tabela I- Resultados A*.

Situação	Tempo médio de cálculo (ms)	Tamanho do caminho (mm)
Interceptar passe	0,404	2.694,992
Receber passe	0,365	2.372,960
Voltar à defesa	26,088	13.244,600

Tabela II - Resultados RRT.

Situação	Tempo médio de cálculo (ms)	Tamanho do caminho (mm)
Interceptar passe	2,736	2.786,236
Receber passe	2,883	2.678,987
Voltar à defesa	25,232	7.427,869

Na Figura 2 pode-se ver o resultado do cálculo do trajeto na situação 1, o caminho em vermelho foi gerado pelo A*, o caminho em preto foi gerado pelo RRT depois da simplificação, a árvore do RRT está em verde e o caminho em cinza foi gerado pelo RRT originalmente.

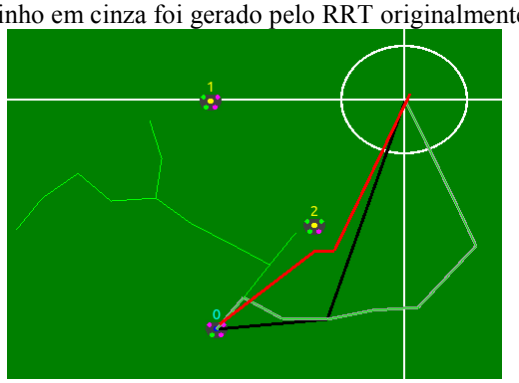

Figura 2- Trajetos calculados utilizando o A* e o RRT.

Nas Figuras 3 e 4 têm-se os gráficos que representam o desempenho do A* e RRT, respectivamente, na situação 1. Nestes gráficos é possível notar a grande variação de tempo de cálculo do RRT em relação ao A*.

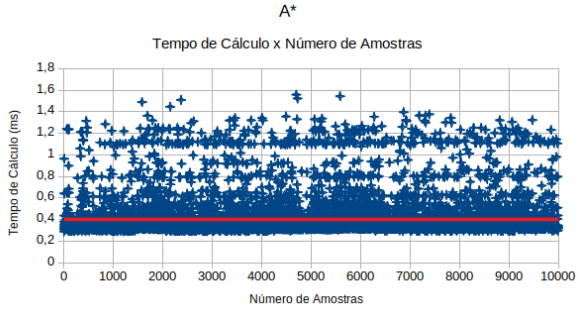

Figura 3 - Gráfico do desempenho do A*.

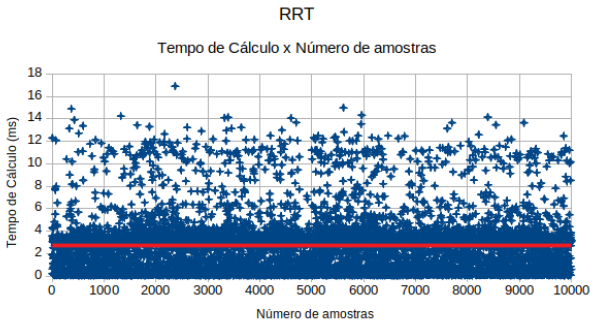

Figura 4 - Gráfico do desempenho do RRT.

6. Conclusões

Analisando os dados coletados comprovou-se que o A* além de possuir o tempo computacional inferior, esse tempo varia pouco em torno da média, ao contrário do RRT. Nas Figuras 3 e 4 é possível notar que a amplitude dos tempos para o A* vale aproximadamente $1{,}4\ ms$ e para o RRT o valor sobe para cerca de $17\ ms$, isso mostra que o A* possui pouca variação no tempo de cálculo para encontrar o mesmo caminho, isso não ocorre no RRT. O tempo de cálculo é importante pois, no momento os times são formados por seis a oito robôs, logo, os tempos obtidos devem ser multiplicados pela quantidade de robôs em campo. De acordo com as Tabelas I e II, o A* também é capaz de gerar o menor caminho possível para chegar ao destino, isto influencia em situações onde deseja-se interceptar um passe, posicionar-se rapidamente a fim de receber um passe ou então voltar para a área de defesa.

7. Referências

[1] HART, Peter E; NILSSON, Nils J; RAPHAEL, Bertram. A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, IEEE, v. 4, n. 2, p. 100–107, 1968.

[2] LAVALLE, Steven M. Rapidly-exploring random trees: A new tool for path planning. Citeseer, 1998.

[3] RULES, Small Size League. Página das regras da SSL. Acesso em 17 ago. 2017 as 10:25. Set. 2011. Disponível em: < https://bit.ly/2LJKqo8 >.

[4] ZICKLER, Stefan et al. CMDragons 2009 extended team description. In: PROC. 14th International RoboCup Symposium, Singapore. [S.l.: s.n.], 2010.

[5] MONAJJEMI, Valiallah; KOOCHAKZADEH, Ali; GHIDARY, Saeed Shiry. grsim–robocup small size robot soccer simulator. In: SPRINGER. ROBOT Soccer World Cup. [S.l.: s.n.], 2011. p. 450–460.

Agradecimentos

À instituição FEI e à equipe RoboFEI pela realização das medidas e empréstimo do laboratório para testes.

[1] Aluno de IC do Centro Universitário FEI (CNPq). Projeto com vigência de 08/17 a 08/18.