ARTIGOS PATH PLANNING

- IMPLEMENTAÇÃO E COMPARAÇÃO DE ALGORITMOS DE PATH PLANNING PARA FUTEBOL DE ROBÔS - CENTRO UNIVERSITÁRIO FEI - LEONARDO DA SILVA COSTA

Este artigo consiste na comparação entre os algoritmos RRT e A*, o ambiente é modelado a partir do espaço de estados com grade regular, foram desenvolvidos em C++ na IDE do QtCreator e o grSim como simulador.

RRT foi implementado com otimizações para suavidade. Para decidir qual o melhor algoritmo entre os dois, porém, após feita esta mudança o desempenho do RRT se equiparou ao do A* e em alguns casos ele chegou a ser um pouco melhor.

A partir da análise dos dados foi possível comprovar que o A* é o algoritmo mais eficiente em comparação ao RRT, devido ao seu baixo tempo de cálculo e caminhos curtos, que são variáveis importantes no futebol de robôs.

- FAST PATH PLANNING ALGORITHM FOR THE ROBOCUP SMALL SIZE LEAGUE - UNIVERSIDAD SANTO TOMÁS, COLOMBIA - SAITH RODRÍGUEZ, EYBERTH ROJAS, KATHERÍN PÉREZ, JORGE LÓPEZ , CARLOS QUINTERO AND JUAN CALDERÓN

Algoritmo usado pela equipe STOx's na Robocup, o desempenho do algoritmo foi analisado usando métricas como a suavidade das trajetórias, a distância percorrida e o tempo de processamento, e comparado com o algoritmo RRT.

Pseudo-código:

```
function FastPathPlanning (environment,trajectory,depth)
obstacle = trajectory.Collides(environment)
if theres is an obstacle and depth < max_recursive then
{
    subgoal=SearchPoint(trajectory,obstacle,environment);
    trajectory1=GenerateSegment(trajectory.start,subgoal);
    trajectory1=FastPathPlanning(environment,trajectory1,depth+1);
    trajectory2=GenerateSegment(subgoal,trajectory.goal);
```

trajectory2=FastPathPlanning(environment,trajectory2,depth+1);

trajectory=JoinSegments(trajectory1,trajectory2);

}

return trajectory;

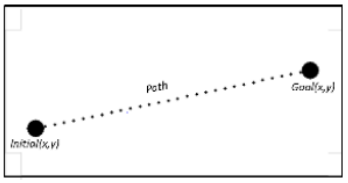
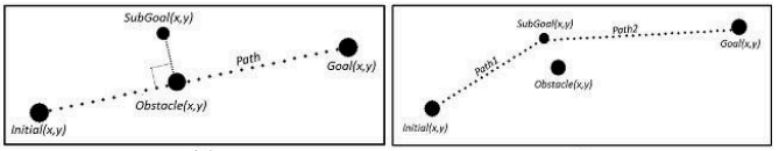


Fig. 3. Path without obstacles



1. Caso de comparação 1: cenário estático.

CASE 1, EVALUATION				
ALGORITHM	TIME(p)	SMOOTH(p)	DIST(p)	EVAL(p)
RRT	0.116	0.712	0.564	0.385
PROPOSAL	0.013	0.066	0.431	0.113

2. Caso de comparação 2: cenário dinâmico, dados coletados e um segmento de uma partida da RoboCup 2013 (STOx's vs CMDragons). Há 281 cenários contínuos no total.

CASE 1, EVALUATION				
ALGORITHM	TIME(p)	SMOOTH(p)	DIST(p)	EVAL(p)
RRT	0.222	0.412	0.880	0.410
PROPOSAL	0.042	0.024	0.712	0.171

Conclusão: Propusemos uma heurística ad-hoc para o planejamento de caminhos em ambientes dinâmicos não desorganizados. Testamos nossa abordagem em um conjunto de situações artificiais e reais da Liga RoboCup Small Size e descobrimos que ela encontra caminhos mais suaves e curtos mais rapidamente no caso médio. É evidente que o bom desempenho do algoritmo

proposto está relacionado às restrições do ambiente onde o aplicamos.. Ao testar a heurística proposta, descobrimos que ela supera a versão implementada pelo RRT em aproximadamente 30%, em média, usando as métricas propostas. Além disso, também observamos que a proposta supera o RRT em todas as três métricas individuais, ou seja, comprimento do caminho, suavidade e tempo de processamento.

- PARTITION HEURISTIC RRT ALGORITHM OF PATH PLANNING BASED ON Q-LEARNING - OMNIROLL (GUANGDONG) INTELLIGENT MANUFACTURING CO., LTD OF LIAONING PROVINCE SHENYANG, CHINA - ZHIYONG LIU ,FEI LAN ,HAIBO YANG.

Para resolver o problema de alta aleatoriedade no planejamento de caminhos tradicional de árvore aleatória (RRT) de exploração rápida, propõe-se um algoritmo de planejamento de RRT heurístico de partição baseado em Q-learning (Q-PRRT).

Os resultados da simulação mostram que o algoritmo Q-PRRT garante a otimalidade do caminho de planejamento global, obtém um caminho de planejamento mais suave e também melhora a eficiência da busca de caminhos e a capacidade de evitar obstáculos. Ele apresenta melhor adaptabilidade em diferentes ambientes de obstáculos.

Pseudo-código:

Q-PRRT Algorithm

Input: Initialize random tree;

Create initial position sinit, target node sgoal, step size η;

Algorithm main loop:

while (||snew-sgoal||>λ)

Find the nearest node to the target node snearest;

Obstacle detection: dF、dB、dL、dR;

while(||Δ||>θ or n>N)

Execute a according to partition heuristic rules;

Generate sampling node s';

if obstacle_free(snew , snearest)

Update Q-value: 1 ,nearest nearest nearest new nearestQ Q r x a Q Q

else

return step 8;

end

end

Select the action with the max Q-value;

Generate new node snew;

Add snew to the tree;

return step 4;

end

Output: tree and path planning

Algorithm		*Uniform obstacle*	*Narrow channel*	*Trap barrier*
Expanding nodes	RRT	4318	2840	3944
	GoalBias-RRT	438	327	705
	Q-PRRT	325	218	588
Expanding time	RRT	13.13	12.97	14.51
	GoalBias-RRT	1.66	1.52	3.45
	Q-PRRT	1.45	1.24	2.24
Path lentgh	RRT	168.06	180.19	185.11
	GoalBias-RRT	153.68	163.53	179.28
	Q-PRRT	136.63	156.06	165.04

- SIMULATING PATH-PLANNING ALGORITHMS USED IN ROBOCUP SSL COMPETITIONS - KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS - AHMED ALHARBI, HAMZA ALSHARIF, ABDULRAHMAN JAVAID E DR KHALED ALSHEHRI.

Comparação entre o STOx's planning, A* e RRT, usando grSim.

O resultado aqui é que o cálculo do STOx está na faixa de microssegundos, enquanto o RRT está na faixa de milissegundos, o A* foi o com pior resultado.