Pathfinder

Degree programme : BSc in Computer Science | Specialisation : Distributed Systems and IoT
Thesis advisor : Prof. Dr. Reto Koenig
Expert : Adrian Egli (SBB AG Informatik Bern)

The Pathfinder solves a problem of the HFTM's "RoboCup" - team: The task of the team's robots is to autonomously find a path between two points within a complex factory set up. As this is done against competing robots, the path has to be found fast. The common shortest path algorithms such as Dijkstra and A* are too slow while maintaining the required precision.

Randomized Algorithms

Common ways for finding a path between two points is to use Dijkstra's or A* algorithm. While these algorithms yield the shortest path, they may be slow especially in large maps with high resolution. For this reason, the Pathfinder bases path planning on randomized algorithms (for example, (informed) optimal rapidly-exploring random trees (RRT*), batch informed trees (BIT*) and others). They don't necessarily find the shortest path, but usually a good path in much less calculation time. The randomized algorithms work by sampling random points and connecting them, typically generating a tree structure.

Precision and scalability

The precision of the results does not depend on any chosen resolution, as it calculates intersections between path and obstacles. In the "RoboCup" use case, usually there are only few obstacles. However, the Pathfinder has been implemented to scale logarithmically with obstacle complexity.

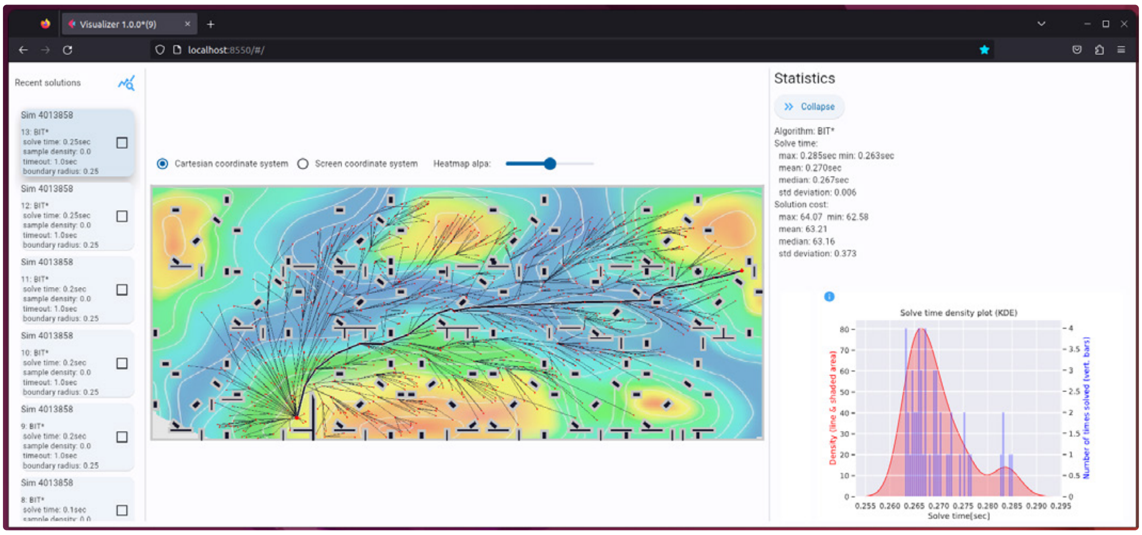
Scalar field / Heatmap

The Pathfinder extends optimal path finding by introducing the notion of a scalar field (heatmap) that can be provided via the API. A given heatmap directly influences the cost function used in order to find the optimal path.

Visualization

Connecting a Visualizer and Simulator (either deployed as web app or native OS app) results in a small distributed system, communicating via an API using the MQTT protocol. The Visualizer can visualize the map with solution path and search tree, and some statistical data about solve time and path cost. The Simulator lets the user test the system and try out different parameters, comfortably using a GUI without having to study and use the API.

Kersten Reinhard Brändle
kersten.braendle@gmail.com



Sim 4013858
13: BIT*
solve time: 0.25sec
sample density: 0.0
timeout: 1.0sec
boundary radius: 0.25 □

Sim 4013858
12: BIT*
solve time: 0.25sec
sample density: 0.0
timeout: 1.0sec
boundary radius: 0.25 □

Sim 4013858
11: BIT*
solve time: 0.2sec
sample density: 0.0
timeout: 1.0sec
boundary radius: 0.25 □

Sim 4013858
10: BIT*
solve time: 0.2sec
sample density: 0.0
timeout: 1.0sec
boundary radius: 0.25 □

Sim 4013858
9: BIT*
solve time: 0.2sec
sample density: 0.0
timeout: 1.0sec
boundary radius: 0.25 □

Sim 4013858
8: BIT*
solve time: 0.1sec
sample density: 0.0 □

● Cartesian coordinate system ○ Screen coordinate system Heatmap alpa:

Statistics

Algorithm: BIT*
Solve time:
max: 0.285sec min: 0.263sec
mean: 0.270sec
median: 0.267sec
std deviation: 0.006
Solution cost:
max: 64.07 min: 62.58
mean: 63.21
median: 63.16
std deviation: 0.373

The Visualizer showing heatmap, search tree, solution path and statistics