Path Planning for Robotics

A brief Review

Zhong Jie

School of Automation

2020 年 5 月 30 日

Outline

Outline

Position and Orientation Representation

1. Pose:position and orientation
2. Frame and coordinate transforms
3. Rotation Representations

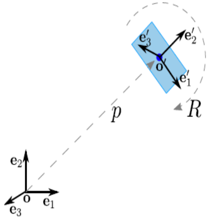


图: position and orientation

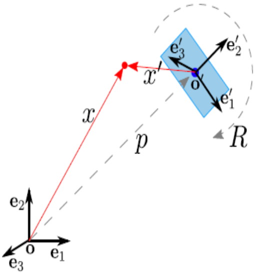
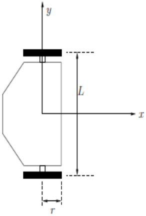


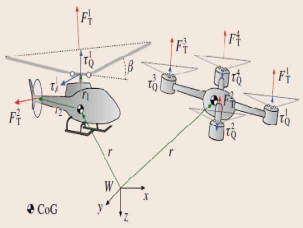
图: frame transforms

Different robots

- Wheel robots



- UAV



- **Manipulation**

Robot Manipulation Kinematics

- DH Table
- Forward Kinematics
- Inverse Kinematics
  - analytical solution
  - numerical solution

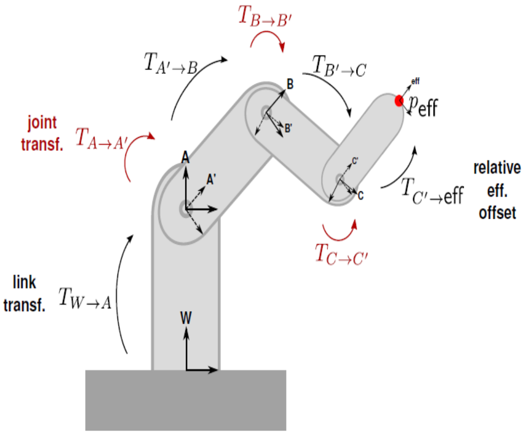


图: model of manipulation

Wheel robot Kinematics

Many mobile robots use a differential drive.

Control method

$$\dot{x} = \frac{r}{2}(u_r + u_l)\cos\theta$$
$$\dot{y} = \frac{r}{2}(u_r + u_l)\sin\theta$$
$$\dot{\theta} = \frac{r}{L}(u_r - u_l)$$

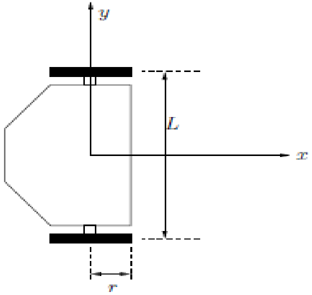


图: mobile robot model

Outline

Path Finding Examples

- Wheel Robot
- UAV

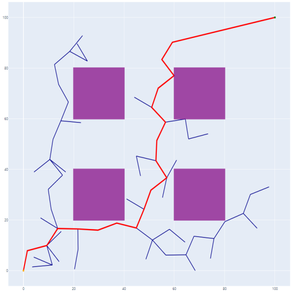


图: path finding in 2D map

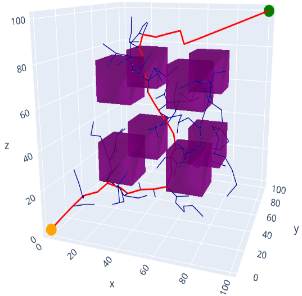


图: path finding in 3D map

Code:`https://github.com/motion-planning/rrt-algorithms`

Path Finding Examples

- **Robot Manipulation planning**

attention

Unlike the path finding in 2D or 3D map,for the manipulation with freedom of six or seven,**planning in a 6D or 7D 'map' is much more difficult**.

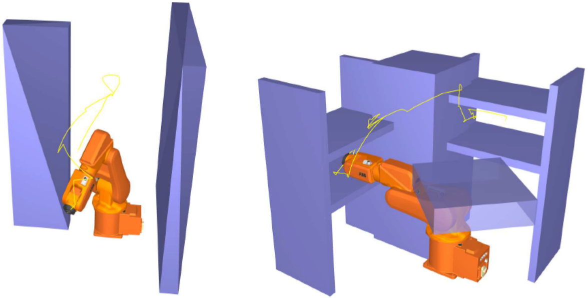


图: path finding in 3D map

motion planning Primitives

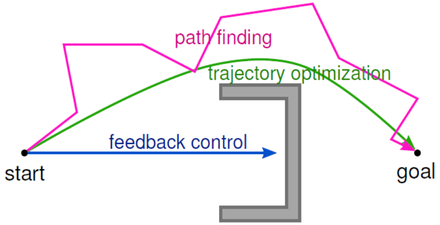


图: Feedback control, path finding, trajectory optim.

- feedback control:E.g.,$q_{t+1} = q_t + J^\sharp(y^* - \phi(q_t))$
- Path Finding Find some q_{0_T} with only valid configurations
- Trajectory Optimization: $\arg\min_{q_{0:T}} f(q_{0:T})$

Outline

"Bugs 2" Algorithm

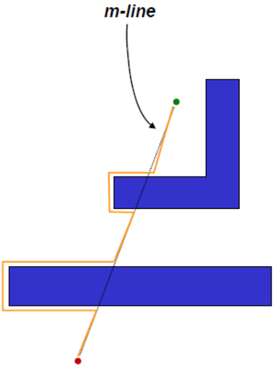


图: bug2 v1.0

- head toward goal on the m-line
- if an obstacle is in the way,follow it until you encounter the m-line again.
- Leave the obstacle and continue toward the goal

"Bugs 2" Algorithm

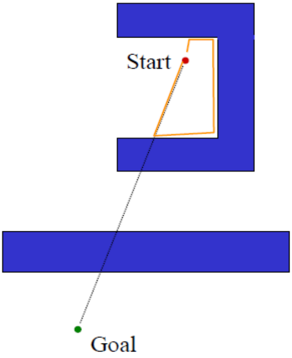


图: bug2 v1.0

BUG!!!

- head toward goal on the m-line
- if an obstacle is in the way,follow it until you encounter the m-line again.
- Leave the obstacle and continue toward the goal

"Bugs 2" Algorithm

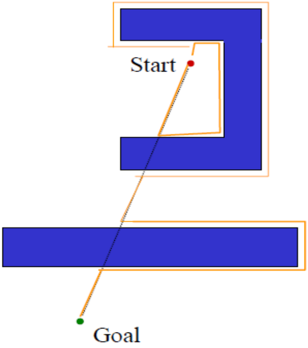


图: bug2 v2.0

- head toward goal on the m-line
- if an obstacle is in the way, follow it until you encounter the m-line again closer to the goal.
- Leave the obstacle and continue toward the goal

"Bugs 2" Algorithm

- **Conclusion**
  - Other variants: TangentBug, VisBug, RoverBug, WedgeBug, . . .
  - only 2D! (TangentBug has extension to 3D)
  - Guaranteed convergence
  - Still active research:
    K. Taylor and S.M. LaValle: I-Bug: An Intensity-Based Bug Algorithm

Tips

Useful for minimalistic, robust 2D goal reaching
not useful for finding paths in joint space

Outline

Artifical Potential Field

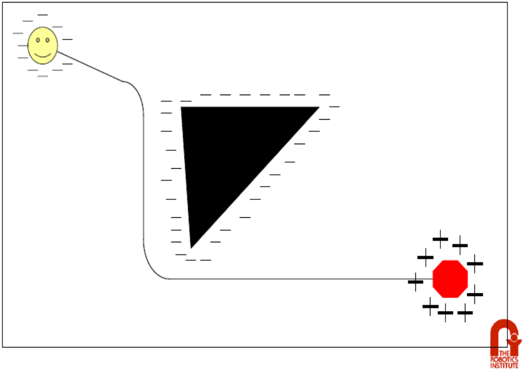


图: Start-Goal Algorithm:Potential Functions

Artifical Potential Field

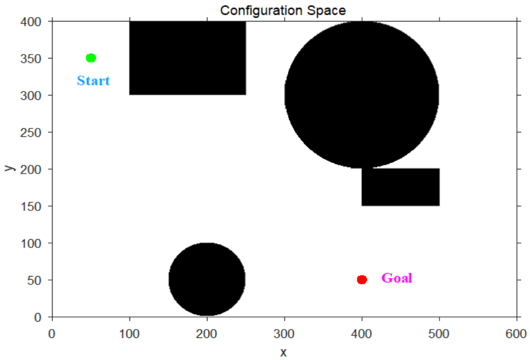


图: A sample

Artifical Potential Field

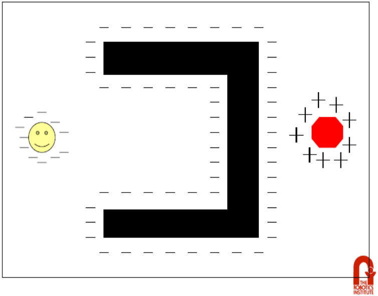


图: Local minimum

- **Conclusion**
  - Very simple , therefore popular
  - Does not solve locality problem of feedback control.

Outline

The Wavefront Action

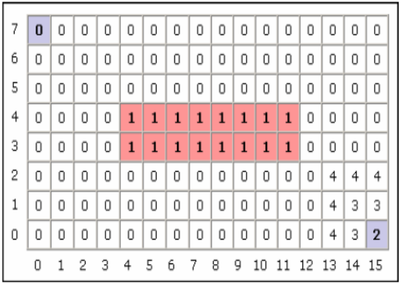


图: start search

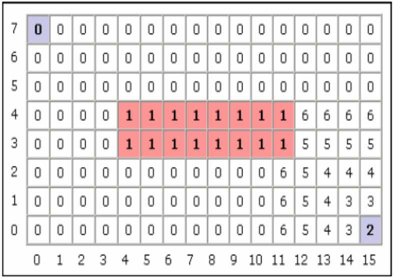


图: and again and again

Starting with the goal, set all adjacent cells with "0" to the current cell +1.We' ll use 8-Point Connectivity in our example

The Wavefront Action Done

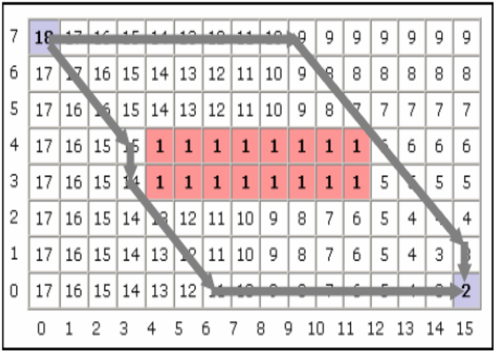


图: path found

To find the shortest path, according to your metric, simply always move toward a cell with a lower number

Dijkstra

- conclusion
  - Is efficient in discrete domains
  - Produces optimal (shortest) paths
  - Applying this to continuous domains requires discretization
    - Grid-like discretization in high-dimensions is daunting! (curse of dimensionality)
    - What are other ways to "discretize" space more efficiently?

A* , a method like Dijkstra,learn about it by yourself if needed!

Outline

Probabilistic Road Maps

Step1:Probabilistic Road Maps –generation

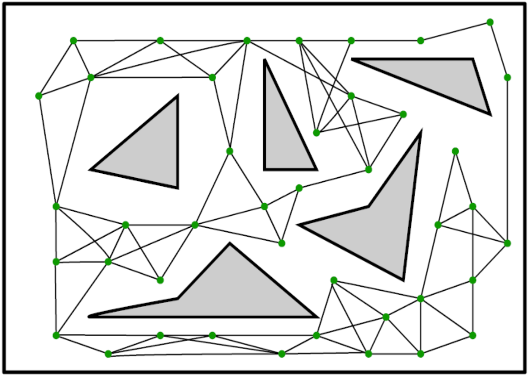


图: generates a graph G = (V;E) of configurations

Probabilistic Road Maps

Step2:Given the graph, use (e.g.) Dijkstra to find path from q_{start} to q_{goal}

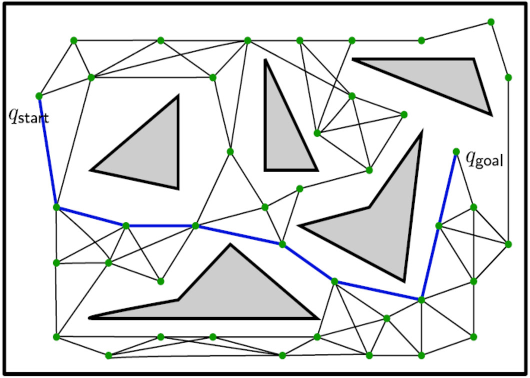


图: slect a path from graph

Probabilistic Road Maps

Conclusion

- Pros
  - Algorithmically very simple
  - Highly explorative
  - Allows probabilistic performance guarantees
  - Good to answer many queries in an unchanged environment
- Cons
  - Precomputation of exhaustive roadmap takes a long time

Outline

Motivations

- Single Query path finding: Focus computational efforts on paths for specific(q_{start}; q_{goal})
- Use actually controllable DoFs to incrementally explore the search space: controlbased path finding.

Tips

Probabilistic Road Maps is a is able to solve different instances of the problem in the same environment,so it is **multi Query**,Planning time is invested in sampling and generating a roadmap

Rapidly Exploring Random Trees family

There are many varietas of RRT algorithm,①more efficiency,② more optimal,③ replanning

- RRT*
- bi-RRT
- inform-RRT
- lazy-RRT
- ...

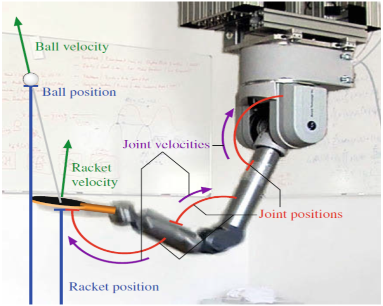
Let's see a sample

RRT VS PRM

- Pros (shared with PRMs):
  - Algorithmically very simple
  - Highly explorative
  - Allows probabilistic performance guarantees
- Pros (beyond PRMs):
  - Focus computation on single query (qstart; qgoal) problem
  - Trees from multiple queries can be merged to a roadmap
  - Can be extended to differential constraints (nonholonomicsystems)
- To keep in mind (shared with PRMs):
  - The metric (for nearest neighbor selection) is sometimes critical
  - The local planner may be non-trivial

Outline

RL = Learning to Act in state



given a state of the robot enviroment

The agent will learn to act with that state for a max final reward. a agent will be representation by a Deep Neural Network,that is the Deep Reiforcement learning.

Applications of RL

- Robotics
  - -Navigation, walking, juggling, helicopters, grasping, etc...
- Game
  - -Backgammon, Chess, Othello, Tetris, ...
- Control
  - –factory processes, resource control in multimedia networks, elevators, ...

Outline

resource

- 研究仿真平台
  - Matlab - Robotics system toolbox and RL toolbox
  - Vrep - robot urdf import and OMPL
  - Gazebo and ROS - a vrep in Linux(ubuntu)
- 算法
  - 编程语言 python matlab
  - 深度强化学习 DRL(我的研究方法)
  - 进化算法 (遗传算法)EA(肇江的研究方法)
  - RRT 系列 (往届师兄研究过)
  - 模型预测控制算法 (感觉很有用的一种算法)

Thanks

一点建议

希望各领域师弟们尽快找到自己的研究点，并在研一上打好一些必要基础。
对硕士学位而言，时间短暂，你需要钻研一种以上的算法以拿证毕业
对于机器人智能规划的师弟，欢迎咨询我们的规划小组师兄，保持密切沟通，尽快开始你们想研究和突破的算法，有问题都可以找师兄们。