比較

RRT : 可以在最短時間&最有效找到路徑，但是沒辦法找到最佳路徑

RRT\* : 可以收斂路徑，但是計算時間趨近於無限

RRT\*-smart : 加入path optimization & intelligent sampling，增加收斂速度

重點內容

RRT

1. 在有限的次數中找到解 -> 不一定是最優
2. 找到新路線不會更新 -> 要更多時間逼近最佳解

RRT\*

1. 尋找最佳解 -> 迭代過程中修改節點
2. 找到新路線會更新 -> 需要更多時間計算、比較

RRT\*-smart

1. 在RRT\*的基礎上優化新節點的選擇

Algorithm

RRT\*

1. InitializeTree : 一個空的tree，紀錄點的運動路徑
2. InsertNode : 將初始節點Zinit插入到tree中，並使其成為root
3. 執行迴圈 :
   1. 隨機抽取一個地方Zrand，代表下一個位置
   2. 找到tree裡面，離Zrand最近的地方Znearest
   3. 計算Znearest移動到Zrand的時間Tnew和狀態Xnew
   4. 檢查Xnew是否碰到障礙區(希望不是)，如果Xnew在障礙區就重找一個Zrand
   5. 離Znew(Zrand)的距離為|V|內找一個鄰近節點Znear
   6. 在Znear裡面找一個父節點Zmin
   7. 把Znew插入到tree裡面，成為Zmin的子節點
   8. 檢查是否有更好的Zmin，有的話就改
4. Return

RRT\*-smart

1. InitializeTree : 一個空的tree，紀錄點的運動路徑
2. InsertNode : 將初始節點Zinit插入到tree中，並使其成為root
3. 執行迴圈 :
   1. 根據迭代次數選擇Zrand(某些迭代會從之前生成、在優化路徑上的點Zbeacons中選擇) -> intelligent sampling
   2. 找到tree裡面，離Zrand最近的地方Znearest
   3. 計算Znearest移動到Zrand的時間Tnew和狀態Xnew
   4. 檢查Xnew是否碰到障礙區(希望不是)，如果Xnew在障礙區就重找一個Zrand
   5. 離Znew(Zrand)的距離為|V|內找一個鄰近節點Znear
   6. 在Znear裡面找一個父節點Zmin
   7. 把Znew插入到tree裡面，成為Zmin的子節點
   8. 檢查是否有更好的Zmin，有的話就改
   9. 如果找到初始路徑，更新n為i(智能抽樣的控制，開始使用偏向抽樣)
   10. 使用PathOptimization直接計算成本directcost
   11. 新的成本較低 --> 更新Zbeacons
4. Return

優化內容

1. Path Optimization
2. 根據三角不等式 c < a+b -> 0到2的距離會比0到1再到2的距離短
3. 一直延伸，直到0碰到了N時碰到了障礙物
4. 經過優化所剩下的節點被稱為Beacons，作為Intelligent Sampling的基礎
5. Intelligent Sampling
6. Intelligent Sampling的核心概念是讓節點盡可能接近障礙物的頂點
7. 上方的Beacons大致上都是需要轉彎而出現，代表可能碰到頂點或是邊
8. 透過以Zbeacons為中心，Rbeacons為半徑，在周遭生成sample，有機會可以快速找到繞過頂點的路

小結

1. RRT\*-smart透過Path Optimization來優化最短路徑
2. Intelligent Sampling可以在一定的迭代次數時生成特定的Zrand，縮短繞過頂點需要的尋找次數