# matlab中：

T.v生成点，T.v(1)为起点，包括节点坐标、父节点坐标、节点与父节点间成本

得到完整路径节点后：再放T\_LIST里，T LIST一行一个节点，5列，列包括节点横纵坐标、父节点横纵坐标、节点索引，第一行为起点

T\_LIST放path里，path一行一个路径坐标，第一行为终点，2列，为xy坐标

参数：

达到终点阈值：Thr

扩展步长：Delta

初始化搜索半径：initial\_radius

半径增加步长：delta\_radius

（安全距离：safeDistance）

（3D图网格的行列数：grid\_size）

碰撞逃逸增幅因子：a

（新点卡障碍物中的次数：n（变量不是参数））

将地图划分为网格：gridSize

死胡同区域封闭阈值：deadEndThreshold

# ros中：

node类型为坐标、节点id、父节点id、起点到节点成本

p rand是std::pair<double, double>

p new在steer函数生成时是std::pair<double, double>但马上赋值给节点newnode

node\_nearest是节点

找完路径最后getPathFromTree函数中：node转化为point并存储于path，path中的点依次转换为pose后存储于plan

最后将plan放入path\_pose，最后在plan话题下发布path\_pose

参数：

达到终点阈值：goal\_radius\_

搜索半径：search\_radius\_（选择父节点与rewire时）

最大迭代次数：max\_nodes\_num\_

规划超时：plan\_time\_out\_

节点之间的最小允许距离：epsilon\_min\_（生长点时）

节点之间的最大允许距离（步长）：epsilon\_max\_（生长点时）

优化：路径点间隔：path\_point\_spacing\_

优化：转角：angle\_difference\_

Node为结构体，包含 x; y; node\_id; parent\_id; cost;

nodes是树，即Node的集合std::vector< Node > nodes;

nodes.push\_back(start\_node);树最后加入新节点

nodes.size();树中节点数量

nodes[i].x 第i个节点x坐标

nodes.back() 树中最后一个节点

# 函数对应：

feasiblePoint(point,map)(无障碍返回true)对应collision(double x, double y)（有障碍返回true）

collisionChecking(startPose,goalPose,map)（无障碍返回true）对应isLineFree(const std::pair<double, double> p1, const std::pair<double, double> p2)（无障碍返回true）

isInCircle(center, point, radius)（在则返回true）对应isInCircle(const std::pair<double, double> p1,const std::pair<double, double> p2,double radius)（在则返回true）

findClosestPointToObstacle(startPoint, endPoint, map, stepSize)对应findClosestPointToObstacle(const std::pair<int, int>& startPoint, const std::pair<int, int>& endPoint, double stepSize)

isSafeDistance(point1, point2, image, safeDist)（是则返回true）对应isLineSafeDistance(const std::pair<double, double> p1, const std::pair<double, double> p2)（是则返回true）