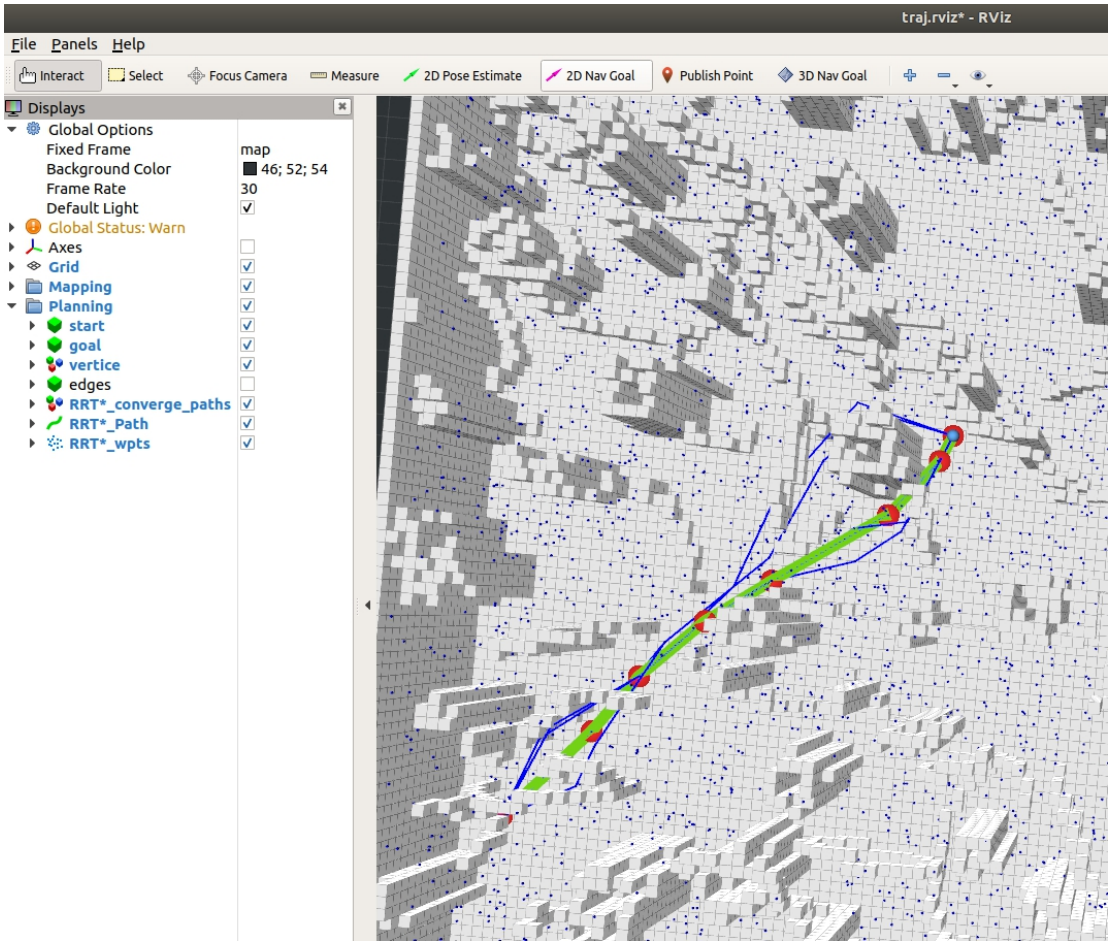
HomeWork 3

一、成果展示




```
------------------------------
goal rcved at -5.05608 -2.88584     0.82
[ INFO] [1657790193.533747435]: [RRT*]: RRT starts planning a path
[ WARN] [1657790193.546053823]: ********************
[ WARN] [1657790193.560184197]: ********************
[ INFO] [1657790193.734858444]: [RRT*]: first path length: 29.7919, use_time
: 0.002153
[ INFO] [1657790193.734918392]: [RRT*] final path len: 27.5039
```

二、代码分析

1. ChooseParent()选择父节点

```
// TODO Choose a parent according to potential cost-from-start values
// ! Hints:
// !  1. Use map_ptr_->isSegmentValid(p1, p2) to check line edge validity;
// !  2. Default parent is [nearest_node];
// !  3. Store your chosen parent-node-pointer, the according cost-from-parent and cost-from-start
// !     in [min_node], [cost_from_p], and [min_dist_from_start], respectively;
// !  4. [Optional] You can sort the potential parents first in increasing order by cost-from-start value;
// !  5. [Optional] You can store the collison-checking results for later usage in the Rewire procedure.
// ! Implement your own code inside the following loop
for (RRTNode3DPtr &curr_node : neighbour_nodes)
{
  if (!map_ptr_->isSegmentValid(curr_node->x, x_new))
  {
    continue;
  }
  double dist2nbr = calDist(curr_node->x, x_new);
  double dist_from_start=dist2nbr+curr_node->cost_from_start;
  if (dist_from_start<min_dist_from_start)
  {
    min_node=curr_node;
    cost_from_p=dist2nbr;
    min_dist_from_start=dist_from_start;
  }
```

首先判断点线的可行性，是否与障碍物碰撞。与 RRT 不同的是选择父节点时，搜索范围内多个节点，在其中选择最小 cost 的节点作为父节点。

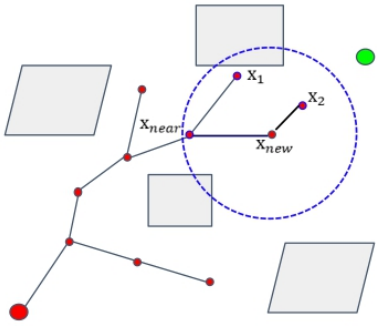
2. Rewire()剪枝

```
        /* 3.rewire */
        // TODO Rewire according to potential cost-from-start values
        // ! Hints:
        // !  1. Use map_ptr_->isSegmentValid(p1, p2) to check line edge validity;
        // !  2. Use changeNodeParent(node, parent, cost_from_parent) to change a node's parent;
        // !  3. the variable [new_node] is the pointer of X_new;
        // !  4. [Optional] You can test whether the node is promising before checking edge collison.
        // ! Implement your own code between the dash lines [--------------] in the following loop
//rewire:遍历x new的邻近节点，查看它们通过x new节点到达起点的路径，是不是比它们之前的路径(dist)要短，如果是则将x new更新为它们的新父节点。
        for (RRTNode3DPtr &curr_node : neighbour_nodes)
        {
          double best_cost_before_rewire = goal_node_->cost_from_start;
          // ! ------------------------------------
          if (!map_ptr_->isSegmentValid(curr_node->x, x_new))
          {
            continue;
          }
          double dist_new_2_nbr = calDist(curr_node->x, x_new);


          if (curr_node->cost_from_start > dist_new_2_nbr + new_node->cost_from_start)
          {
            changeNodeParent(curr_node,new_node,dist_new_2_nbr);

          }

          // ! ------------------------------------
          if (best_cost_before_rewire > goal_node_->cost_from_start)
          {
            ROS_WARN("********************");
            vector<Eigen::Vector3d> curr_best_path;
            fillPath(goal_node_, curr_best_path);
            path_list_.emplace_back(curr_best_path);
            solution_cost_time_pair_list_.emplace_back(goal_node_->cost_from_start, (ros::Time::now() - rrt_start_time).toSec());
          }
        }
        /* end of rewire */
      }
```

rewire()遍历 x_{new} 的邻近节点，查看它们通过 x_{new} 节点到达起点的路径，是不是比它们之前的路径要短，如果是则将 x_{new} 更新为它们的新父节点。

三、分析

RRT* 是 RRT 算法的改进版本，主要是增加了剪枝的过程，通过剪枝，RRT* 是概率完备并且概率最优的，但是 RRT* 仍需要在整个状态空间内进行采样。



Algorithm 2: RRT Algorithm

Input: $\mathcal{M}, x_{init}, x_{goal}$
Result: A path Γ from x_{init} to x_{goal}
$\mathcal{T}$.init();
for $i = 1$ *to* n **do**
  $x_{rand} \leftarrow Sample(\mathcal{M})$;
  $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
  $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
  if *CollisionFree*(x_{new}) **then**
    $X_{near} \leftarrow NearC(\mathcal{T}, x_{new})$;
    $x_{min} \leftarrow ChooseParent(X_{near}, x_{near}, x_{new})$;
    $\mathcal{T}.addNodEdge(x_{min}, x_{new})$;
    $\mathcal{T}.rewire()$;

首先生成 x_{rand} 然后在树中寻找离 x_{rand} 最近的点 x_{near} 最后在线段方向延伸出一段距离作为 x_{new}

与 RRT 不同的是选择父节点时，搜索范围内多个节点，在其中选择最小 cost 的节点作为父节点，本题中范围内有$\{x_1, x_2, x_{near}\}$三个可选节点，其中 x_{new} 经过 x_{near} 到达起始点的 cost 的值最小，所以选择 x_{near} 为其父节点。这是与 RRT 不同的。

rewire()函数的作用是 更改节点的父节点，剪枝的意思，可以使得节点到达起始点的 cost 变小。这使得轨迹持续的优化(找到轨迹，轨迹不断优化，这是有别于 RRT 的改进)。

使用 **Kd-tree** 改进寻找最近节点 x_{near} 的函数，提高效率。

四、心得

首先理解代码逻辑时，要先了解代码中的数据结构，这对理解代码有着很重要的作用。

其次通过这次作业，让我认识到快速解决问题的能力还是有待提升的，在代码调试代码的过程中如何快速的找出问题所在，这个的完成作业的过程让我意识到 **debug** 的重要性，和培养 **debug** 能力是解决问题的利器。我在调试代码的时候，会陷入自己的思维误区，会自以为是，认为代码是这样或那样运行的，但实际上一些运行的程序并不是你想的那样，所以遇到自己检查很久，调试很久也没找到问题所在的，那么就 **debug** 一下吧，看流程是不是与这项的那样，这样可以定位问题所在。

最后就是要与人多沟通交流，有时候自己想了很久想不明白的东西，他人的一句话就可能使我们醒悟，和他人多沟通一方面可以提升技术，理解更深，而且可以培养表达能力，有时候我们工科生并不注重自己的与人沟通的表达能力的培养，只专注于技术本身。