Project4: Kinodynamic & OBVP

控制空间采样和状态空间采样

控制空间采样将输入（jerk，加速度等）离散化后进行积分得到对应的状态，积分过程中可以判断是否有冲突。

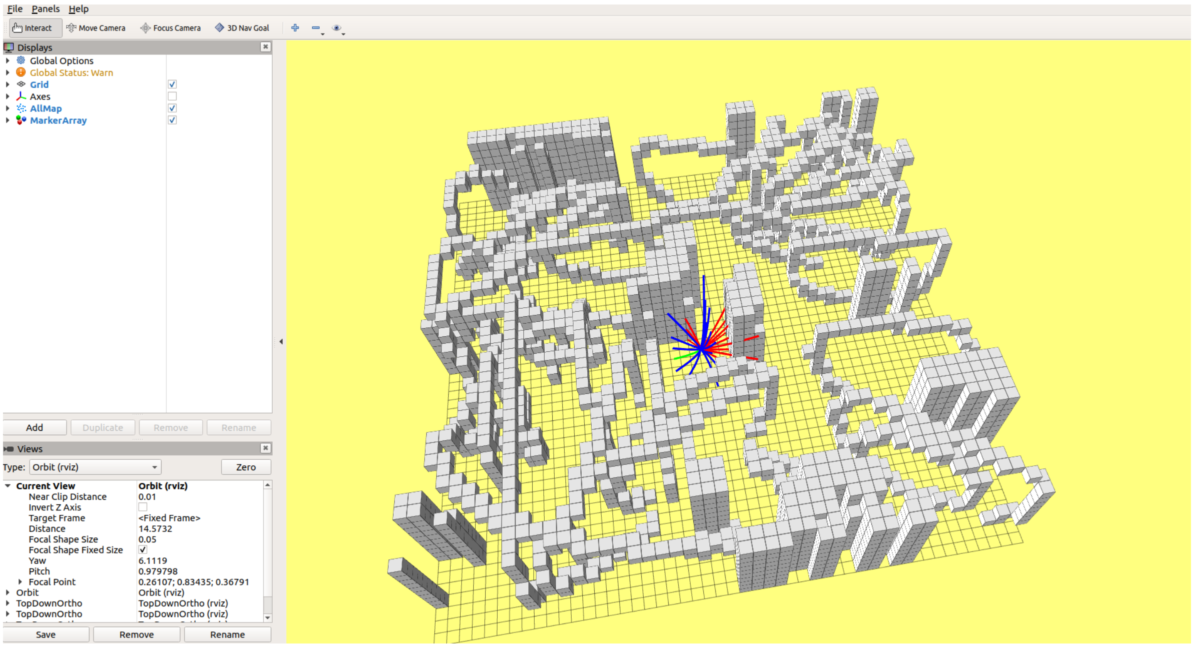
```
for (int step = 0; step <= _time_step; step++)
        {
          double total_time = delta_time * step;
          pos(0) = start_pt(0) + start_velocity(0) * total_time + 0.5 * acc_input(0) * total_time * total_time;
          pos(1) = start_pt(1) + start_velocity(1) * total_time + 0.5 * acc_input(1) * total_time * total_time;
          pos(2) = start_pt(2) + start_velocity(2) * total_time + 0.5 * acc_input(2) * total_time * total_time;
          vel(0) = start_velocity(0) + acc_input(0) * total_time;
          vel(1) = start_velocity(1) + acc_input(1) * total_time;
          vel(2) = start_velocity(2) + acc_input(2) * total_time;


          Position.push_back(pos);
          Velocity.push_back(vel);
          double coord_x = pos(0);
          double coord_y = pos(1);
          double coord_z = pos(2);
          // check if if the trajectory face the obstacle
          if (_homework_tool->isObsFree(coord_x, coord_y, coord_z) != 1)
          {
            collision = true;
          }
        }
```

状态空间采样给定始末状态，利用minimum principle构建costate求解对应代价最小的路径。过程中会用到求多项式方程的根，可以构造多项式的伴随矩阵求特征值来实现：

```
double Homeworktool::OptimalBVP(Eigen::Vector3d _start_position, Eigen::Vector3d _start_velocity,
Eigen::Vector3d _target_position)
{
  double optimal_cost = 100000;
  std::vector<double> obvp_cost;
  Eigen::Vector3d delta_p = _target_position - _start_position;
  Eigen::Vector3d v0 = _start_velocity;

  double c0 = -36.0 * (delta_p[0] * delta_p[0] + delta_p[1] * delta_p[1] + delta_p[2] * delta_p[2]);
  double c1 = 24.0 * (delta_p[0] * v0[0] + delta_p[1] * v0[1] + delta_p[2] * v0[2]);
  double c2 = -4.0 * v0.norm() * v0.norm();
  double c3 = 0.0;

  Eigen::Matrix<double, 4, 4> companionMatrix;
  Eigen::Matrix<complex<double>, Eigen::Dynamic, Eigen::Dynamic> companionEigenvalues;
  companionMatrix << 0, 0, 0, -c0,
    1, 0, 0, -c1,
    0, 1, 0, -c2,
    0, 0, 1, -c3;
  companionEigenvalues = companionMatrix.eigenvalues();
  Eigen::Vector3d a;
  Eigen::Vector3d b;

  for (int i = 0; i < companionEigenvalues.size(); i++)
  {
    if (companionEigenvalues(i).real() > 0.0 && companionEigenvalues(i).imag() < 1e-12)
    {
      double t = companionEigenvalues(i).real();
      for (int j = 0; j < 3; j++)
      {
        a(j) = -6.0 / std::pow(t, 3) * (2 * delta_p[j] - v0[j] * t);
        b(j) = 2.0 / std::pow(t, 2) * (3 * delta_p[j] - 2.0 * v0[j] * t);
      }
      double coef3 = a.norm() * a.norm() / 3.0;
      double coef2 = a(0) * b(0) + a(1) * b(1) + a(2) * b(2);
      double coef1 = b.norm() * b.norm();
      double cost = t + coef3 * std::pow(t, 3) + coef2 * std::pow(t, 2) + coef1 * t;
      obvp_cost.emplace_back(cost);
    }
  }
  if (!obvp_cost.empty())
  {
    optimal_cost = *max_element(obvp_cost.begin(), obvp_cost.end());
  }
  return optimal_cost;
}
```

demo示例

红色为碰撞路径，蓝色为可行路径，绿色为optimal_cost路径。



Global Options
Global Status: Warn
Grid ☑
Axes □
AllMap ☑
MarkerArray ☑

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip Distance	0.01
Invert Z Axis	□
Target Frame	<Fixed Frame>
Distance	14.5732
Focal Shape Size	0.05
Focal Shape Fixed Size	☑
Yaw	6.1119
Pitch	0.979798
Focal Point	0.26107; 0.83435; 0.36791
Orbit	Orbit (rviz)
TopDownOrtho	TopDownOrtho (rviz)
TopDownOrtho	TopDownOrtho (rviz)

To Do

看Kinodynamic RRT*和hyber A*