第4章作业

1.作业说明

- 运行环境：ROS2 Humble
- 代码详见`src`目录，修改较大，将原始代码移植到ROS2平台中，主要修改的代码在`grid_path_searcher`包中的`hw_tool.hpp/cpp`，`demo2_node.hpp/cpp`中。

主要实现内容：

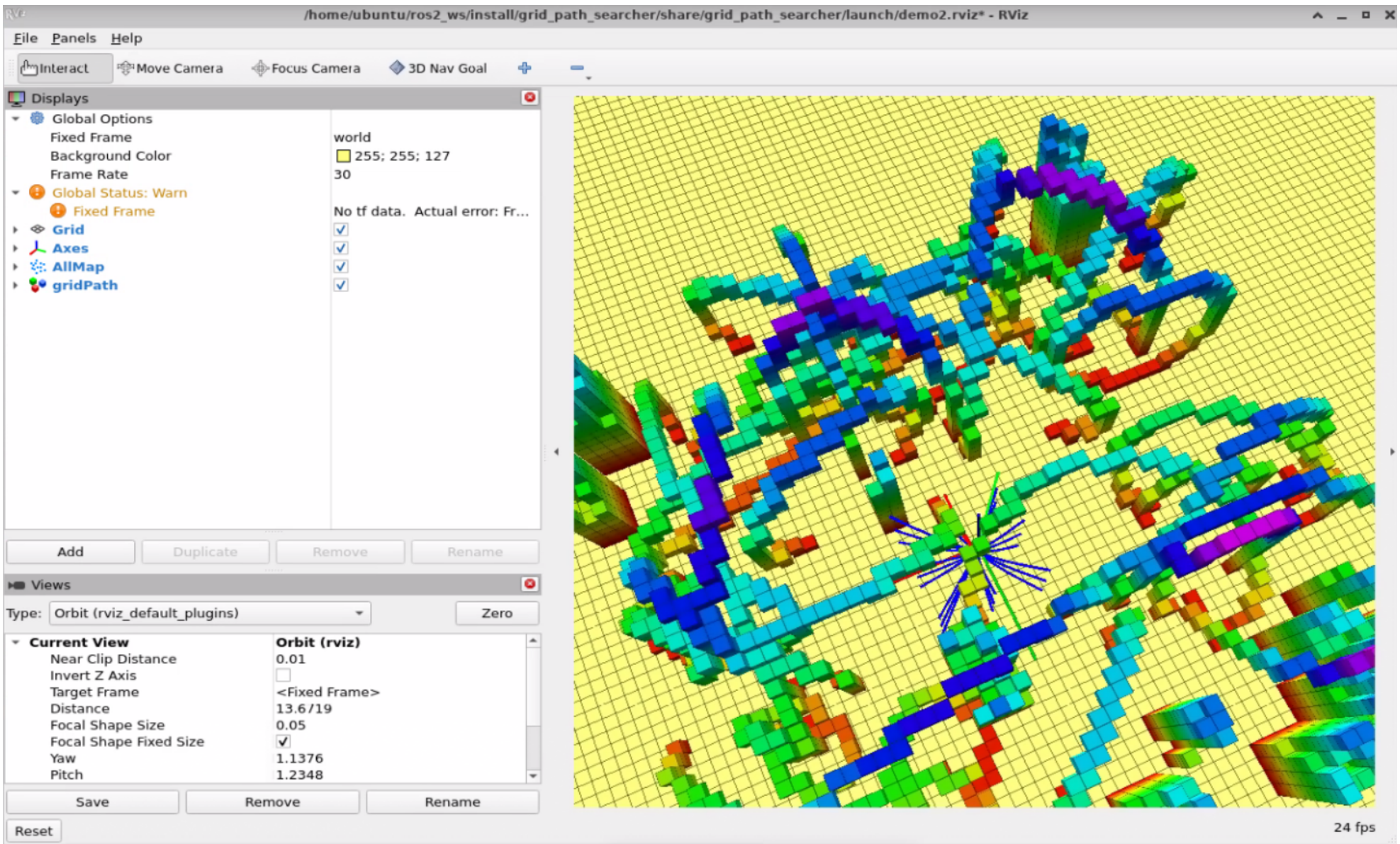
- 在`demo2_node.cpp`中，实现了前向运动积分。
- 在`hw_tool.cpp`中，基于ceres solver实现了`OBVP`问题的数值近似求解。

2.作业运行结果

运行命令

```
ros2 launch grid_path_searcher demo2.launch.py
```

随机在地图上选点，得到rviz可视化运行结果如下图：



3.OBVP求解

详见`hw_tool.cpp`中

定义优化问题：

```
struct MyFunc {
  MyFunc(Eigen::Vector3d &p0, Eigen::Vector3d &pf, Eigen::Vector3d &v0, Eigen::Vector3d &vf)
      : p0_(p0), pf_(pf), v0_(v0), vf_(vf) {}
```

```
4
5   template <typename T>
6   bool operator()(const T* const x,
7                   T* residuals) const {
8     auto t = x[0];
9     auto t2 = t*t;
10    auto t3 = t2*t;
11    auto dp = pf_ - v0_ * t - p0_;
12    auto dv = vf_ - v0_;
13    auto alpha = -12.0 / t3 * dp + 6.0 / t2 * dv;
14    auto beta =  6.0 / t2 *  dp - 2.0 / t * dv;
15    auto J = t;
16    for(int i = 0; i<3; i++) {
17        J += alpha(i)*alpha(i)*t3 + alpha(i)*beta(i)*t2 + beta(i)*beta(i)*t;
18    }
19    residuals[0] = J;
20    return true;
21  }
22
23    // Factory to hide the construction of the CostFunction object from
24    // the client code.
25   static ceres::CostFunction* Create(Eigen::Vector3d &p0, Eigen::Vector3d &pf, Eigen::Vector3d &v0, Eigen::Vector3d
    &vf) {
26     return (new ceres::AutoDiffCostFunction<MyFunc, 1, 1>(new MyFunc(p0, pf, v0, vf)));
27   }
28
29   Eigen::Vector3d p0_;
30   Eigen::Vector3d pf_;
31   Eigen::Vector3d v0_;
32   Eigen::Vector3d vf_;
33  };
34
```

求解优化问题：

```
1     // Build the problem.
2     ceres::Problem problem;
3     ceres::CostFunction* cost_function = MyFunc::Create(_start_position, _start_velocity, _target_position,
    target_velocity);
4     problem.AddResidualBlock(cost_function, nullptr, &x);
5     problem.SetParameterLowerBound(&x, 0, 0.001);
6
7     // Run the solver!
8     ceres::Solver::Options options;
9     options.linear_solver_type = ceres::DENSE_QR;
10    options.minimizer_progress_to_stdout = false;
11    ceres::Solver::Summary summary;
12    ceres::Solve(options, &problem, &summary);
```

补充说明：

- 我们的问题是最小化代价$J = \int_0^T (1 + a_x^2 + a_y^2 + a_z^2) dt$，从`J`的定义中可以发现$J > 0$，因此等价于最小化$J^2$，因此可以利用Ceres进行优化求解，同时设定$T > 0.001$，避免得到无意义的负值。