Trajectory planning

1. Start

Start MoveIT

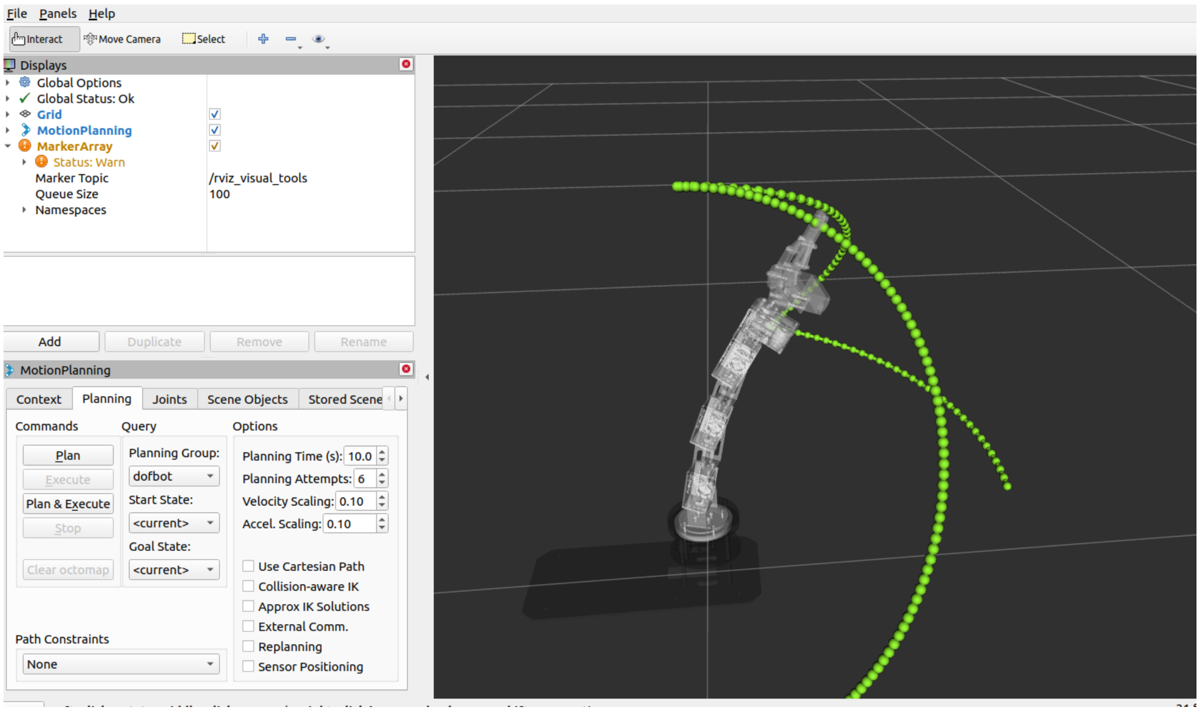
```
roslaunch dofbot_config demo.launch
```

Start trajectory planning node

```
rosrun dofbot_moveit 06_multi_track_motion
```

renderings

To view the trajectory, you need to add the [MarkerArray] plug-in and select the [/rviz_visual_tools] topic.



Global Status: Ok
Status: Warn
Marker Topic /rviz_visual_tools
Queue Size 100
Planning Group: dofbot
Start State: <current>
Goal State: <current>
Planning Time (s): 10.0
Planning Attempts: 6
Velocity Scaling: 0.10
Accel. Scaling: 0.10
Path Constraints: None

Given three reachable target points of the robotic arm, MoveIT will plan three feasible trajectories based on the target points, and then merge the three trajectories into one continuous trajectory.

2. Source code analysis

Set three reachable target points (you can have several target points, they must be reachable)

```
    vector<vector<double>> poses{
            {1.34, -1.0, -0.61, 0.2, 0},
            {0, 0, 0, 0, 0},
            {-1.16, -0.97, -0.81, -0.79, 3.14}
    };
    for (int i = 0; i < poses.size(); ++i) {
        multi_trajectory(yahboomcar, poses.at(i), trajectory);
    }
```

Plan each trajectory

```
void multi_trajectory(
         moveit::planning_interface::MoveGroupInterface &yahboomcar,
         const vector<double> &pose, moveit_msgs::RobotTrajectory &trajectory) {
     moveit::planning_interface::MoveGroupInterface::Plan plan;
     const robot_state::JointModelGroup *joint_model_group;
     // Get the starting position of the robot
     moveit::core::RobotStatePtr start_state(yahboomcar.getCurrentState());
     joint_model_group = start_state->getJointModelGroup(yahboomcar.getName());
     yahboomcar.setJointValueTarget(pose);
     yahboomcar.plan(plan);
     start_state->setJointGroupPositions(joint_model_group, pose);
     yahboomcar.setStartState(*start_state);
     trajectory.joint_trajectory.joint_names =
plan.trajectory_.joint_trajectory.joint_names;
     for (size_t j = 0; j < plan.trajectory_.joint_trajectory.points.size(); j++)
{

trajectory.joint_trajectory.points.push_back(plan.trajectory_.joint_trajectory.po
ints[j]);
     }
}
```

Trajectory merge

```
     moveit::planning_interface::MoveGroupInterface::Plan joinedPlan;
     robot_trajectory::RobotTrajectory rt(yahboomcar.getCurrentState()-
>getRobotModel(), "arm_group");
     rt.setRobotTrajectoryMsg(*yahboomcar.getCurrentState(), trajectory);
     trajectory_processing::IterativeParabolicTimeParameterization iptp;
     iptp.computeTimeStamps(rt, 1, 1);
     rt.getRobotTrajectoryMsg(trajectory);
     joinedPlan.trajectory_ = trajectory;
```

Track display

```
     moveit_visual_tools::MoveItVisualTools tool(yahboomcar.getPlanningFrame());
     tool.deleteAllMarkers();
/*
...
*/
// display track
     tool.publishTrajectoryLine(joinedPlan.trajectory_,
yahboomcar.getCurrentState()->getJointModelGroup("arm_group"));
     tool.trigger();
```

Execute trajectory planning

```
     if (!yahboomcar.execute(joinedPlan)) {
         ROS_ERROR("Failed to execute plan");
         return false;
     }
```