高层运动服务接口

介绍

高层控制接口：通过调用 ROS2 Topic/调用 ROS2 Service/调用 C++ 函数，来给 Saturn Robot 发送速度控制、姿态控制等运动指令。

功能	C++	ROS
上下使能	✓	
控制移动方向和速度	✓	✓
调整站立高度	✓	✓
设置当前运行场景	✓	✓
演示模式控制	✓	
调节运行速度	✓	
软件急停	✓	
恢复软件急停	✓	
设置控制模式	✓	✓

1. 上下使能

1.1. C++ 接口

`rcClientInterfaceDriverEnable` 简介

```
/**
 * @brief 控制机器人上下使能接口。
 *
 * 该函数用于启用或禁用机器人的驱动。
 *
 * @param from 指定操作模式（导航或操纵杆）。
 * @param driver_enable 指定启用（true）或禁用（false）驱动。
 * @return bool 返回 `true` 表示操作成功，返回 `false` 表示操作失败。
 */
bool rcClientInterfaceDriverEnable(robot_control::common::NAV_OR_JOY_MODE
from, bool driver_enable);
```

使用案例

```cpp
robot_control::common::NAV_OR_JOY_MODE mode = robot_control::common::NAV_OR_JOY_MODE::joy_control;
bool enable = true;
bool success = rcClientInterfaceDriverEnable(mode, enable);
if (success) {
    std::cout << "Driver Enable Command Successful." << std::endl;
} else {
    std::cout << "Failed to Enable Driver." << std::endl;
}
```

2. 控制移动方向与速度

2.1. C++接口

`rcClientInterfaceDirectionMovement`

```cpp
/**
 * @brief 设置机器人运动方向与运动速度的指令(前后左右以及转向)。
 *
 * 该函数用于控制机器人的运动方向和速度。
 *
 * @param from 指定操作模式（导航或操纵杆）。
 * @param rc_direct 指定机器人的运动方向和速度。
 * @return bool 返回 `true` 表示操作成功，返回 `false` 表示操作失败。
 */
bool rcClientInterfaceDirectionMovement(robot_control::common::NAV_OR_JOY_MODE from,
            common::ROBOT_TWIST rc_direct, int64_t time_millis);
```

使用案例

```cpp
robot_control::common::NAV_OR_JOY_MODE mode = robot_control::common::NAV_OR_JOY_MODE::joy_control;
// 获取机器人状态
common::ROBOT_COMMON_STATUS status;
rcClientInterfaceGetCommonStatus(&status);
// 设置以0.5m/s的速度往前行走(示例值)
common::ROBOT_TWIST direction = {{0.5, 0.0, 0.0}, {0.0, 0.0, 0.0}};
bool success = rcClientInterfaceDirectionMovement(mode, direction, status.heartbeat);
if (success) {
    std::cout << "Direction Movement Command Successful." << std::endl;
} else {
    std::cout << "Failed to Command Direction Movement." << std::endl;
}

注意：完整案例代码请参考应用开发模块中的创建客户应用章节。
```

2.2. ROS2 接口

该接口用于订阅 ROS2 的 `/cmd_vel` topic，以获取机器人运动的速度指令。

Topic 名称	Topic 类型	角色
`/cmd_vel`	`geometry_msgs::msg::Twist`	订阅方

消息结构

`geometry_msgs::msg::Twist` 包括以下字段：

- `linear`：包含 `x`，`y`，`z`，表示线速度。
- `angular`：包含 `x`，`y`，`z`，表示角速度。

示例消息

```cpp
geometry_msgs::msg::Twist twist_msg;
twist_msg.linear.x = 0.1;
twist_msg.linear.y = 0.0;
twist_msg.linear.z = 0.0;
twist_msg.angular.x = 0.0;
twist_msg.angular.y = 0.0;
twist_msg.angular.z = 0.0;
```

测试方法

- 发布一次性消息 使用以下命令发布一次性消息到 `/cmd_vel` topic：

```
ros2 topic pub --once /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.1, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
```

- 持续发布消息
  使用以下命令以每秒10次的频率持续发布消息到 `/cmd_vel` topic：

```
ros2 topic pub --rate 10 /cmd_vel geometry_msgs/msg/Twist "{linear: {x: 0.2, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}"
```

注意事项

- 确保ROS2节点已启动并正在订阅 `/cmd_vel` topic。
- 发布的消息格式必须严格符合 `geometry_msgs::msg::Twist` 消息类型的定义。
- 持续发布消息时，注意控制发布频率，避免网络拥塞。

3. 调节站立高度

3.1. C++接口

`rcClientInterfaceBodyHighAdjust` 简介

```
/**
 * @brief 调节机器人的站立高度的接口。
 *
 * 该函数用于调节机器人的站立高度。
 *
 * @param from 指定操作模式（导航或操纵杆）。
 * @param scale 指定调节高度的比例，正值表示增加高度，负值表示降低高度。
 * @return bool 返回 `true` 表示调节成功，返回 `false` 表示调节失败。
 */
bool rcClientInterfaceBodyHighAdjust(robot_control::common::NAV_OR_JOY_MODE from, int scale);
```

使用案例

```cpp
robot_control::common::NAV_OR_JOY_MODE mode = robot_control::common::NAV_OR_JOY_MODE::joy_control;
int height_adjustment = 10; // 示例值，表示增加高度
bool success = rcClientInterfaceBodyHighAdjust(mode, height_adjustment);
if (success) {
    std::cout << "Body Height Adjustment Successful." << std::endl;
} else {
    std::cout << "Failed to Adjust Body Height." << std::endl;
}
```

注意事项

- 确保传入的操作模式和高度调整比例height_adjustment是0到100的范围内的整数。

3.2. ROS2接口

该服务用于调整机器人的身体高度。

Service 名称	Service 类型	角色
`/saturn/rc_height_scale`	`saturn_msgs::srv::HeightScale`	客户端

请求消息

`saturn_msgs::srv::HeightScale::Request` 消息类型包含以下字段：

- `scale`：整数类型，表示高度比例

示例消息

```cpp
saturn_msgs::srv::HeightScale::Request request;
request.scale = 8;

saturn_msgs::srv::HeightScale::Response response;
```

测试方法

```bash
ros2 service call /saturn/rc_height_scale saturn_msgs/srv/HeightScale "{scale: 8}"
```

4. 设置当前运行场景

4.1. C++接口

`rcClientInterfaceSetScene` 简介

```cpp
/**
 * @brief 设置机器人当前运行场景的接口（站立，趴下，楼梯）。
 *
 * 该函数用于设置机器人的当前运行场景。
 *
 * @param from 指定操作模式（导航或操纵杆）。
 * @param scene 指定场景类型（站立、趴下、楼梯）。
 * @return bool 返回 `true` 表示设置成功，返回 `false` 表示设置失败。
 */
bool rcClientInterfaceSetScene(robot_control::common::NAV_OR_JOY_MODE from, common::SCENE_TYPE scene);
```

使用案例

```cpp
robot_control::common::NAV_OR_JOY_MODE mode = robot_control::common::NAV_OR_JOY_MODE::joy_control;
common::SCENE_TYPE scene = common::WALKING; // 示例值，表示站立场景
bool success = rcClientInterfaceSetScene(mode, scene);
if (success) {
    std::cout << "Scene Set Successfully." << std::endl;
} else {
    std::cout << "Failed to Set Scene." << std::endl;
}
```

4.2. ROS2 接口

该服务用于调整机器人的身体高度。

Service 名称	Service 类型	角色
`/saturn/robot_command`	`ysc_robot_msgs::srv::RobotCommand`	客户端

请求消息

`ysc_robot_msgs::srv::RobotCommand::Request` 消息类型包含以下字段：

- `command_name`：字符串，表示具体命令

命令如下：

- `StandUpDown`：站立/躺下切换
- `EnableNavigation`：启动导航模式
- `EnableTeleoption`：启动手柄模式
- `StairsGait`：切换到楼梯步态
- `OrdinaryGait`：切换到普通步态
- `SlopeGait`：切换到斜坡步态
- `RLOrdinaryGait`：切换到强化学习普通步态
- `RLMountainGait`：切换到强化学习山地步态
- `StayNormal`：站立高度设定到普通高度
- `StayDown`：站立高度设定到较低高度

示例消息

```
saturn_msgs::srv::HeightScale::Request request;
request.scale = "StairsGait"; //楼梯步态

saturn_msgs::srv::HeightScale::Response response;
```

测试方法

```
ros2 service call /saturn/robot_command ysc_robot_msgs/srv/Robot_Command "
{command_name: 'StairsGait'}"
```

5. 演示模式控制

5.1. C++接口

`rcClientInterfaceDemoControl` 简介

```
/**
 * @brief 演示模式控制。
 *
 * 该函数用于控制机器人进入或退出演示模式。
 *
 * @param from 指定操作模式（导航或操纵杆）。
 * @param mode 指定演示模式（2:摇头 3:摆尾 4:招手）
 * @param enable 指定是否开启演示模式（true表示开启，false表示关闭）。
 * @return bool 返回 `true` 表示控制成功，返回 `false` 表示控制失败。
 */
bool rcClientInterfaceDemoControl(
    common::NAV_OR_JOY_MODE from,
    int mode,
    bool enable);
```

使用案例

```
robot_control::common::NAV_OR_JOY_MODE mode = robot_control::common::NAV_OR_JOY_MODE::joy_control;
int demo_mode = 2; // 示例值，表示演示模式
bool enable_demo = true; // 示例值，表示开启演示模式
bool success = rcClientInterfaceDemoControl(mode, demo_mode, enable_demo);
if (success) {
    std::cout << "Demo Mode Control Set Successfully." << std::endl;
} else {
    std::cout << "Failed to Set Demo Mode Control." << std::endl;
}
```

注意事项

- 确保传入的演示模式和使能状态是有效的。

6. 调节运行速度

6.1. C++接口

`rcClientInterfaceSpeedAdjust` 简介

```cpp
/**
 * @brief 调节机器人的运行速度的接口。
 *
 * 该函数用于调节机器人的运行速度。
 *
 * @param from 指定操作模式（导航或操纵杆）。
 * @param scale 指定调节速度的比例（范围 1-100）。
 * @return bool 返回 `true` 表示调节成功，返回 `false` 表示调节失败。
 */
bool rcClientInterfaceSpeedAdjust(robot_control::common::NAV_OR_JOY_MODE from, int scale);
```

使用案例

```cpp
robot_control::common::NAV_OR_JOY_MODE mode = robot_control::common::NAV_OR_JOY_MODE::joy_control;
int speed_adjustment = 5; // 示例值，表示增加速度
bool success = rcClientInterfaceSpeedAdjust(mode, speed_adjustment);
if (success) {
    std::cout << "Speed Adjustment Successful." << std::endl;
} else {
    std::cout << "Failed to Adjust Speed." << std::endl;
}
```

7. 软件急停

7.1. C++接口

`rcClientInterfaceDriverSoftEstop` 简介

```cpp
/**
 * @brief 软件急停接口。
 *
 * 该函数用于触发机器人进行软件急停操作。
 *
 * @param from 指定操作模式（导航或操纵杆）。
 * @return bool 返回 `true` 表示软件急停成功，返回 `false` 表示软件急停失败。
 */
bool rcClientInterfaceDriverSoftEstop(robot_control::common::NAV_OR_JOY_MODE from);
```

使用案例

```cpp
robot_control::common::NAV_OR_JOY_MODE mode = robot_control::common::NAV_OR_JOY_MODE::joy_control;
bool success = rcClientInterfaceDriverSoftEstop(mode);
if (success) {
    std::cout << "Soft Estop Command Successful." << std::endl;
} else {
    std::cout << "Failed to Command Soft Estop." << std::endl;
}
```

8. 恢复软件急停

8.1. C++接口

`rcClientInterfacercResumeSoftEstop` 简介

```cpp
/**
 * @brief 恢复软件急停接口。
 *
 * 该函数用于触发机器人恢复软件急停状态。
 *
 * @param from 指定操作模式（导航或操纵杆）。
 * @return bool 返回 `true` 表示恢复软件急停成功，返回 `false` 表示恢复软件急停失败。
 */
bool rcClientInterfacercResumeSoftEstop(robot_control::common::NAV_OR_JOY_MODE from);
```

使用案例

```cpp
robot_control::common::NAV_OR_JOY_MODE mode = robot_control::common::NAV_OR_JOY_MODE::joy_control;
bool success = rcClientInterfacercResumeSoftEstop(mode);
if (success) {
    std::cout << "Resume Soft Estop Command Successful." << std::endl;
} else {
    std::cout << "Failed to Resume Soft Estop." << std::endl;
}
```

9. 设置控制模式

该服务设置控制机器人的模式，切换为手动或导航模式。

9.1. C++ 接口

`rcClientInterfaceSetNavOrJoyControl` 简介

```
/**
 * @brief 设置导航控制或者遥控控制模式。
 *
 * 该函数用于设置机器人为导航控制模式或遥控控制模式。
 *
 * @param nav_joy 指定控制模式（导航或操纵杆）。
 * @return bool 返回 `true` 表示设置成功，返回 `false` 表示设置失败。
 */
bool rcClientInterfaceSetNavOrJoyControl(common::NAV_OR_JOY_MODE nav_joy);
```

使用案例

```
common::NAV_OR_JOY_MODE mode = robot_control::common::NAV_OR_JOY_MODE::joy_control;
bool success = rcClientInterfaceSetNavOrJoyControl(mode);
if (success) {
    std::cout << "Nav or Joy Control Set Successfully." << std::endl;
} else {
    std::cout << "Failed to Set Nav or Joy Control." << std::endl;
}
```

注意事项

- 确保传入的控制模式是有效的。
- 设置机器人位姿

9.2. ROS2 接口

Service 名称	Service 类型	角色
`/saturn/robot_command`	`ysc_robot_msgs::srv::RobotCommand`	客户端

请求消息

`ysc_robot_msgs::srv::RobotCommand::Request` 消息类型包含以下字段：

- `command_name`：字符串，表示具体命令
  - `EnableNavigation` 表示导航模式
  - `EnableTeleoption` 表示手柄控制模式

示例消息

```
saturn_msgs::srv::HeightScale::Request request;
request.scale = "EnableNavigation"; //导航

saturn_msgs::srv::HeightScale::Response response;
```

测试方法

```
ros2 service call /saturn/rc_nav_or_manual saturn_msgs/srv/NavOrManual "{co
mmand_name: 'EnableNavigation'}"
```