AR 标签跟随功能使用与讲解

1. 功能简介

AR 标签跟随功能是通过 AR 标签识别功能检测 AR 标签并标记 AR 标签的位置后，再驱动小车跟随 AR 标签。

2. 使用方法

在 ROS 中，对于 melodic 版本和 noetic 版本，提供了不同的 AR 标签功能包，ROS-melodic 对应的功能包为 ar-track-alvar，ROS-noetic 对应的功能包为 aruco_ros，因此在后续的内容中，我们将针对不同的 ROS 版本进行讲解。

在【**AR 标签识别功能**】手册中，我们已经讲解过 AR 标签的生成方法，此处不再赘述。

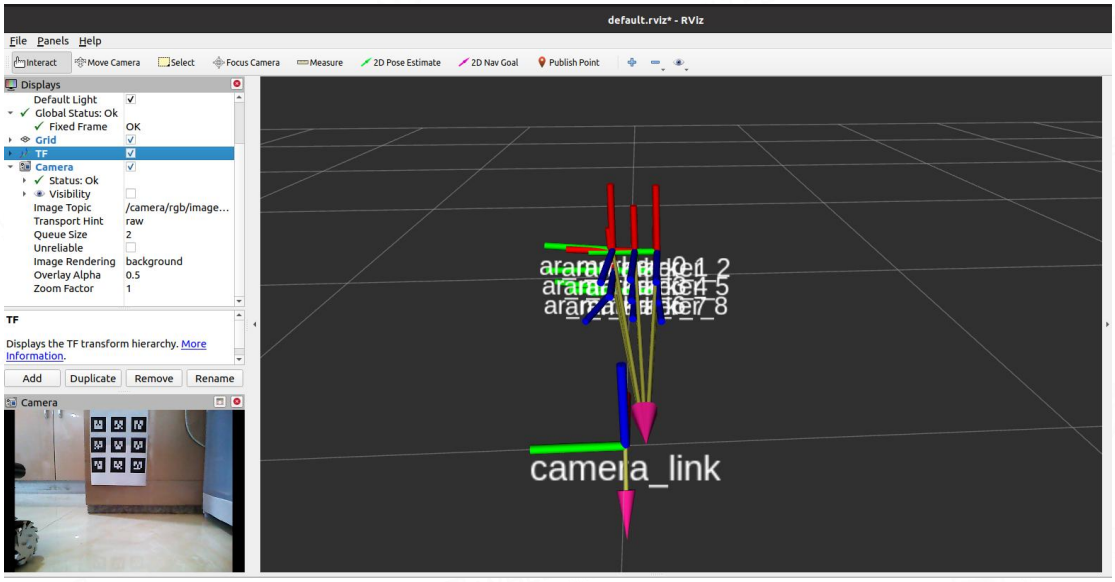
使用前需要确保上位机连接小车 WiFi，并已经 SSH 远程登录到小车端。

① ROS-melodic（Jetsonnano、树莓派等 ROS 主控使用的版本）

1) 在登录后的终端输入启动 AR 标签跟随功能的指令

```
roslaunch simple_follower ar_follower.launch
```

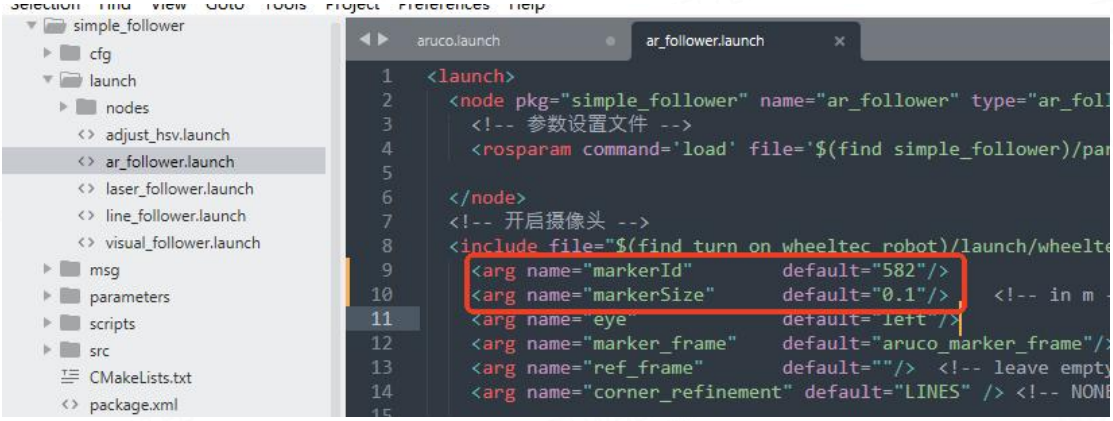
2) 运行 launch 文件后，AR 跟随节点就会被打开，此时在摄像头前方移动需要跟随的 AR 标签，小车就会跟着移动，打开 rviz，基坐标选择 camera_link，如下图所示我们就能在 rviz 中观察到小车摄像头和 AR 标签的 TF 坐标。



小车摄像头和 AR 标签的 TF 坐标

② ROS-noetic（Orin 等主控使用的版本）

1) 在 launch 文件中设置好与所生成标签一致的 id 与 size，源码中默认设置 id 为 582，size 为 0.1m



ar_follower.launch 文件可修改位置

2) 在登录后的终端输入启动 AR 标签跟随功能的指令

```
roslaunch simple_follower ar_follower.launch
```

3) 运行 launch 文件后，AR 跟随节点就会被打开，功能启动完成初始化之后，我们将前面生成的 id 为 582 的 AR 标签放置在小车相机前方，小车将会对 AR 标签进行保持一定距离的跟随

跟随效果示例

同样的，我们也可以参考 AR 识别功能的操作，通过查看话题或者使用可视化工具来查看识别结果

3. 注意事项

① ROS-melodic（Jetsonnano、树莓派等 ROS 主控使用的版本）

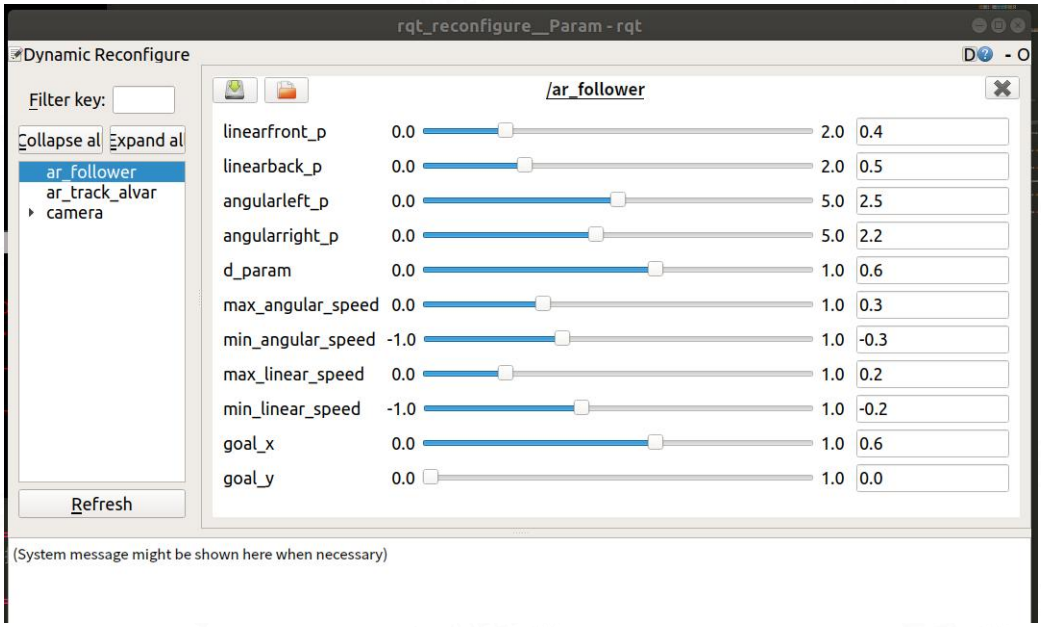
AR 标签跟随功能的动态调参：

在 AR 标签跟随功能中，我们也设置了 rqt 实时动态调参功能，运行命令行

```
rosrun rqt_reconfigure rqt_reconfigure
```

选择 ar_follower，可以看到这里面有很多参数可以进行调节，可调的参数有：

小车向各个方向移动时的速度调节参数、最大最小线速度角速度以及小车与 AR 标签保持的距离和方向。



rqt 动态实时调参窗口

AR 标签跟随功能由于需要确定 AR 标签位姿，因此也具有 AR 标签识别的功能，在 rviz 中的操作方法是一样的，用户可以根据的自己的需求选择想要使用的功能。

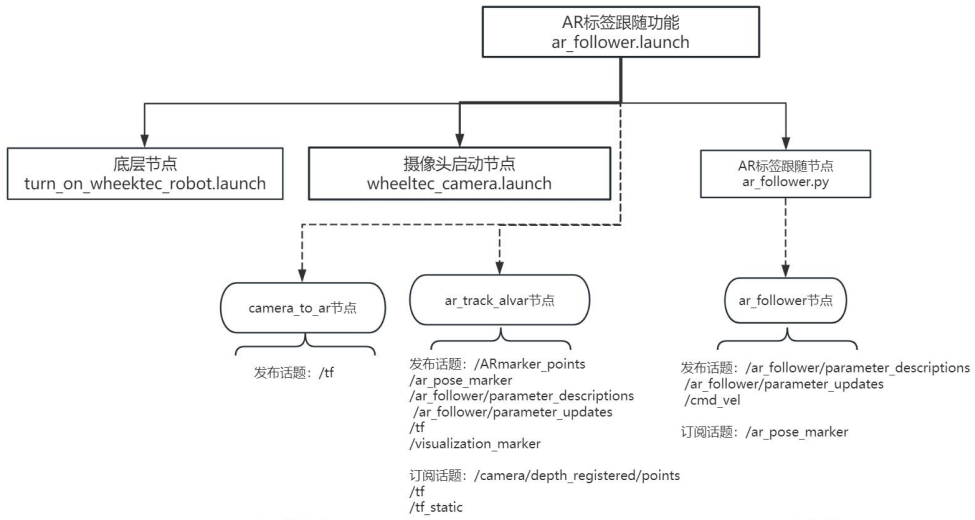
② ROS-noetic（Orin 等主控使用的版本）

由于 aruco_ros 源码包的话题发布机制限制，AR 标签跟随功能在标签离开相机视野后，小车仍会保持原速进行跟随运动，对于 AR 标签跟随功能我们已经做了限速处理防止严重失控，如想小车在看不到标签后停止运动，可以选择先退出功能再移开 AR 标签，或者在跟随运动达到静止状态时快速将 AR 标签移开

4. 功能讲解

AR 标签跟随功能是通过启动文件 ar_follower.launch 来开启的，因 melodic 版本与 noetic 版本在跟随功能上的实现思路一致，此处不再区分版本，而是以 melodic 为例进行功能逻辑讲解。

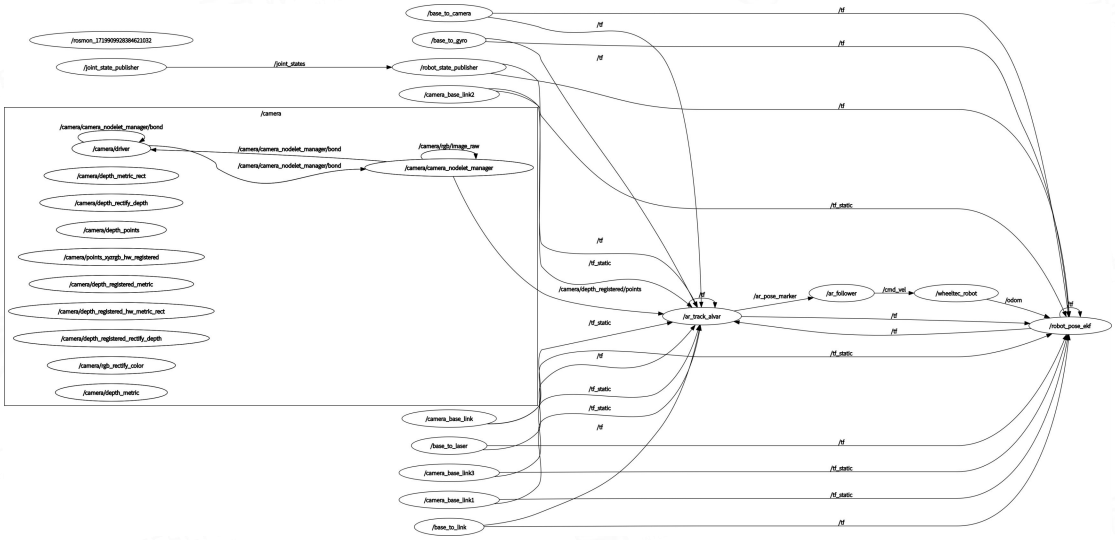
① 启动 AR 标签跟随功能



AR 标签跟随功能启动框架

② AR 标签跟随功能节点关系

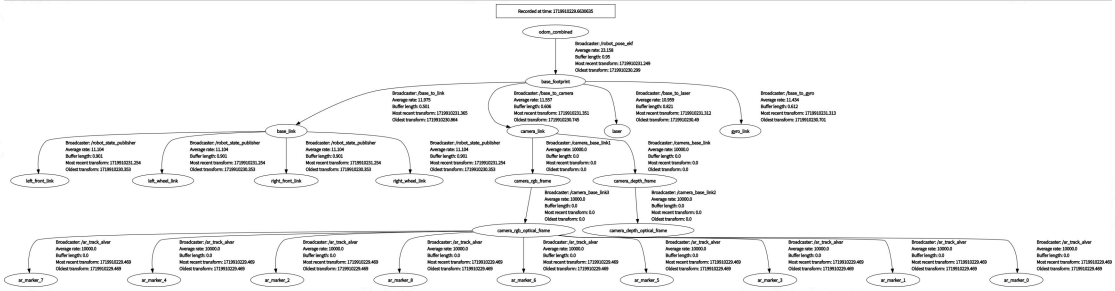
```
rqt_graph
```



AR 标签跟随功能节点关系图

③ AR 标签跟随功能 TF 树

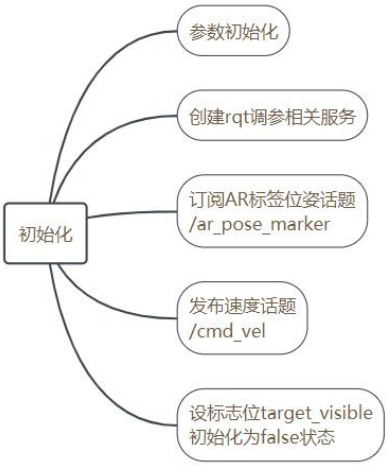
```
rosrun rqt_tf_tree rqt_tf_tree
```



AR 标签跟随功能 TF 树

④ AR 标签跟随功能启动文件介绍

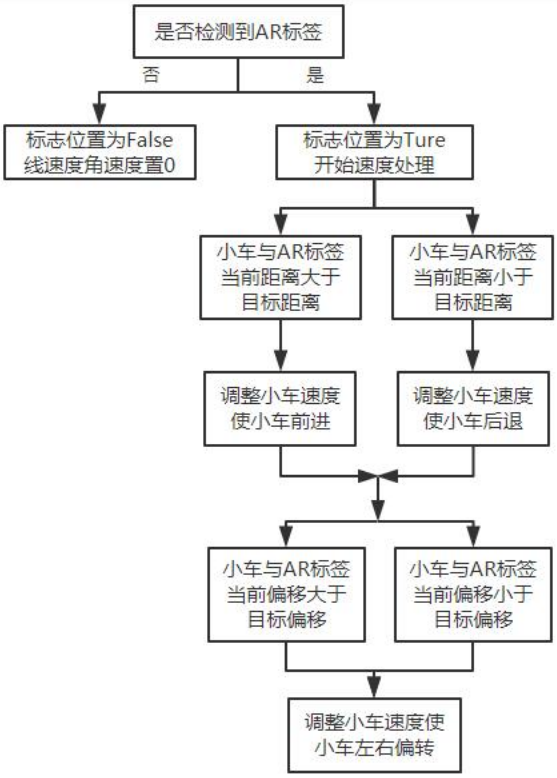
AR 标签跟随功能主要是在 ar_follower.py 文件中实现的，首先是初始化



AR 标签跟随初始化内容

要跟随 AR 标签运动，我们就需要获取 AR 标签的位姿信息，这里我们订阅了话题/ar_pose_marker，它能返回我们所设定的坐标系与 AR 标签之间的距离与偏移。

接下来是关于标签识别和速度处理的程序，设置了一个标志位作为程序判断是否识别到标签的条件，在摄像头识别到标签后对该标志位进行置位，之后根据所获取的位姿信息以及已设置的参数对速度进行处理发布。



AR 标签跟随标签识别和速度处理

```
        #先设定初始化时未检测到 AR 标签
        self.target_visible = False
        self.move_cmd = Twist()

        rospy.loginfo("ar_follow starting!")

        #未检测到进程退出时持续发布速度信息
        while not rospy.is_shutdown():
            self.cmd_vel_pub.publish(self.move_cmd)
            rate.sleep()

    def set_cmd_vel(self, msg):
        try:
            marker = msg.markers[0] #设定以检测到的第一个 AR 标签为目标
            if not self.target_visible:
                rospy.loginfo("the robot is Tracking Target!")
            self.target_visible = True #检测到 AR 标签后将检测标志位置为 ture
        except:
            if self.target_visible:
                rospy.loginfo("Robot Lost Target!")
            self.target_visible = False
            self.move_cmd.linear.x = 0
            self.move_cmd.angular.z = 0
                #失去目标时停下
            return

        offset_y = 0.06 #小车中心与摄像头检测到的 AR 标签中心的偏差
        target_offset_y = marker.pose.pose.position.y + offset_y #AR 标签位姿信
息 y 方向（已校正）
        target_offset_x = marker.pose.pose.position.x #AR 标签位姿信息 x 方向

#注：该部分程序为节选
```