AR 标签识别功能使用与讲解

1. 功能简介

AR 标签识别功能是通过相机识别到相机视野内的 AR 标签，通过相机读取坐标和图像信息等得到 AR 标签与相机之间的 TF 和可视化的标记点。

2. AR 标签功能包安装和介绍

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。

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）

功能包安装直接运行下述的指令即可：

```
sudo apt-get install ros-melodic-ar-track-alvar
```

ar_track_alvar 有 4 个主要功能:

（1）生成不同大小、分辨率和数据/ID 编码的 AR 标签；

（2）识别和跟踪单个 AR 标签的姿态，可选择集成相机深度数据，以更好地估计姿态；

（3）识别和跟踪由多个标签组成的“标签整体”的姿态。这允许更稳定的姿态估计、对遮挡的鲁棒性、对多边目标的跟踪；

（4）使用相机图像自动计算“标签整体”中的标签之间的空间关系，这样用户就不必手动测量并在 XML 文件中输入标签位置来使用“标签整体”功能。

对于该功能包我们采用的是二进制安装方式，无法在我们的工作空间/wheeltec_robot 中直接查看，它的安装路径在/opt/ros/melodic/share/ar_track_alvar，如果想要了解这个功能包可以通过 ROS 的 wiki 页面或者 github 查看它的源代码，ar_track_alvar 的 ROS Wiki 链接——http://wiki.ros.org/ar_track_alvar。

② ROS-noetic（Orin 等主控使用的版本）

aruco_ros 功能包可以在 github 上获取源码：GitHub - pal-robotics/aruco_ros at noetic-devel

aruco_ros 具有以下几个特性：

(1) 可以对 AR 标签进行高帧率跟踪

(2) 可以生成给定大小的 AR 标签，并针对最小感知模糊性进行了优化（当有更多标签需要跟踪时）

(3) 可以通过使用标签板提高跟踪精度

可以应用于目标姿态估计、视觉伺服等。在我们的成品镜像中已将该功能包安装到工作空间，可以在 src 中直接查看，更多关于 aruco_ros 的介绍也可以在 github 页面中获取。

3. 使用方法

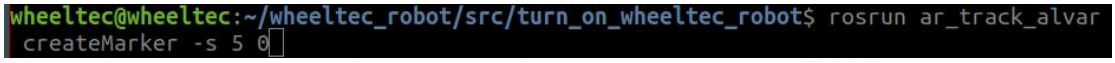
使用前需要确保虚拟机连接小车 WiFi，并已经 SSH 远程登录到小车端。

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

1) 在登录后的终端运行生成识别所需的标签的指令

```
rosrun ar_track_alvar createMarker -s 5 0
```

-s 后面的第一个数字为生成标签的边长，单位为厘米，第二个数字为标签编号。



rosrun ar_track_alvar createMarker -s 5 0

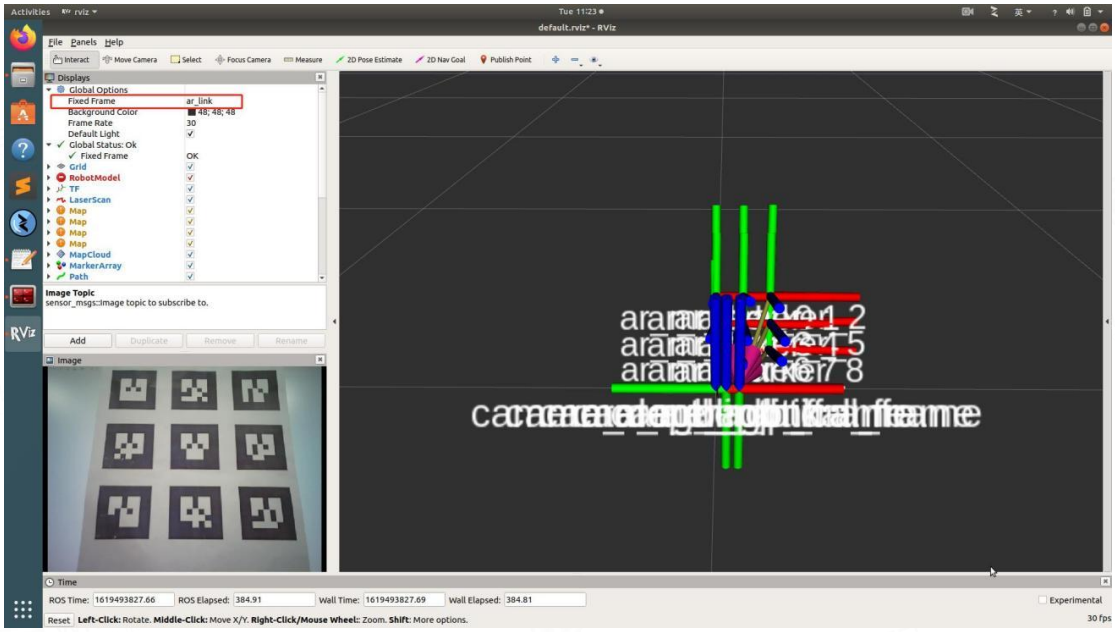
生成的标签保存路径为终端路径，例如上图进入了 turn_on_wheeltec_robot 路径运行，生成的标签也就会保存在这个路径下。识别所需要的标签也可以从 ROS wiki 下载，编号 0-8 的 AR 标签图片链接——http://wiki.ros.org/ar_track_alvar?action=AttachFile&do=view&target=markers0to8.png。

2) 在登录后的终端输入启动 AR 标签识别功能的指令

```
roslaunch turn_on_wheeltec_robot ar_label.launch
```

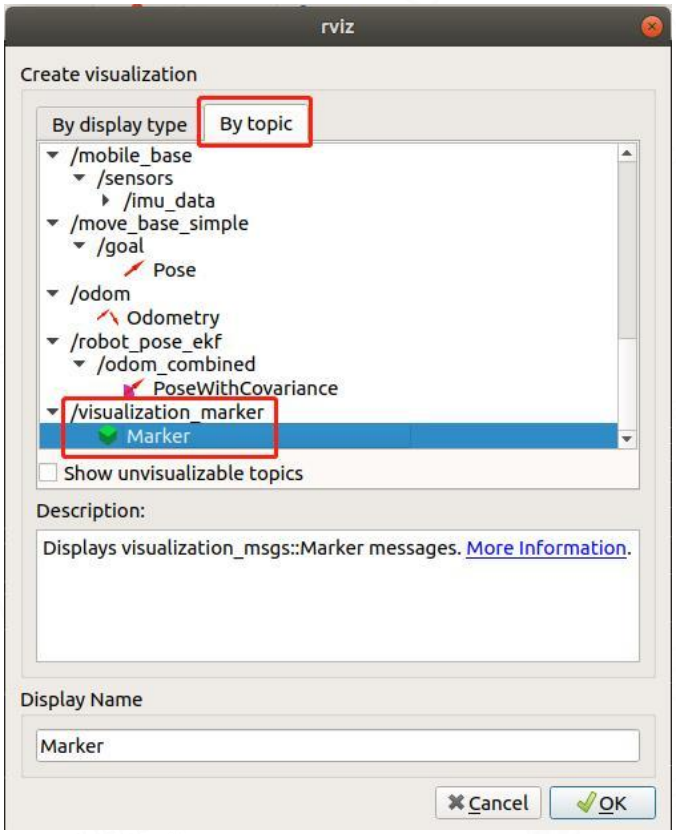
3) 在虚拟机端打开 rviz 可视化工具，查看识别的效果：

Fixed Frame 基坐标选择 ar_link 或 camera_link 都可以，这时我们就可以查看到小车摄像头的 TF 坐标，将标签置于摄像头前，rviz 中就会显示对应 AR 标签的 TF 坐标

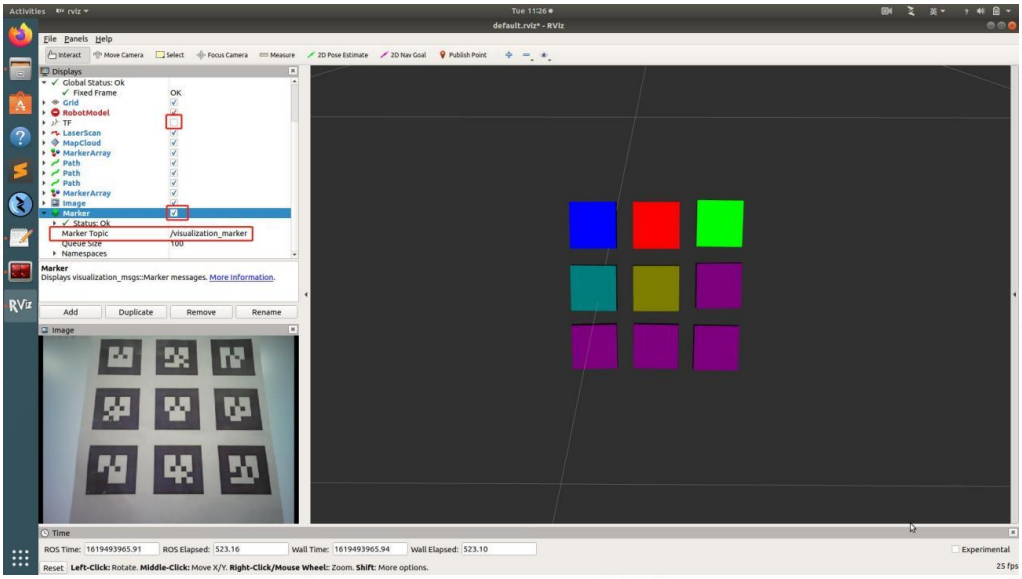


AR 标签识别 rviz 显示效果（带 TF）

AR 标签在 rviz 中还有另外一种显示方式，可以在 rviz 左下方点击 Add，添加一个可查看的话题 marker，话题选择/visualization_marker，然后将 TF 选项勾选去掉，在 rviz 中 AR 标签就会以小方块的形式显示出来。



选择添加 By topic 中/visualization_marker 下的 Marker



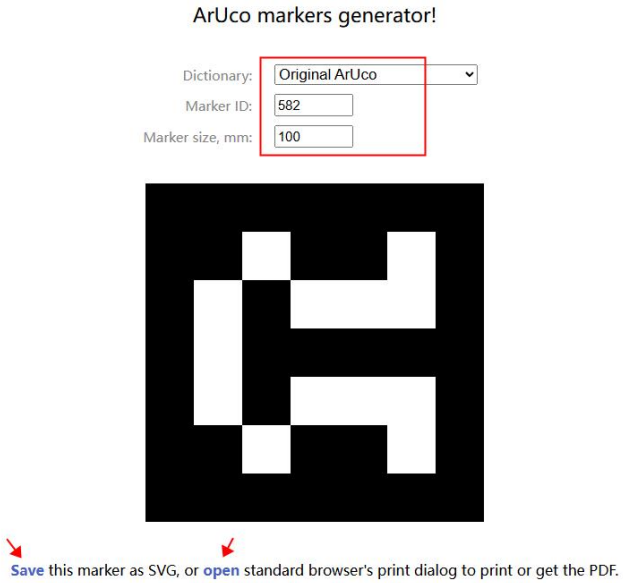
AR 标签识别 rviz 显示效果（不勾选 TF）

注：摄像头识别到 AR 标签时终端会有报错出现，这是因为识别 AR 标签时摄像头无法识别点云信息，可理解为资源被占用，该错误可以忽略，与功能实现无关。

② ROS-noetic（Orin 等主控使用的版本）

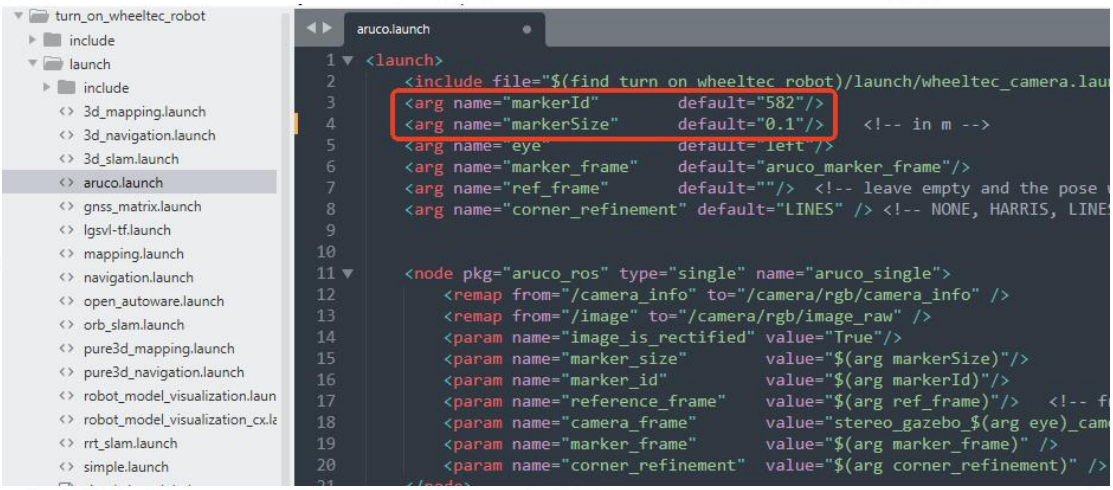
1) 生成 aruco 码供识别使用

aruco 码可以在对应的网页链接中生成：https://chev.me/arucogen/，注意生成 aruco 码时，需要选择 Original Aruco，同时选择与 launch 文件中设定一致的 id 与 size，生成的二维码可以在下方选择进行保存或打印操作



生成 arcuo 码

生成对应的 aruco 码之后，我们还需要在 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



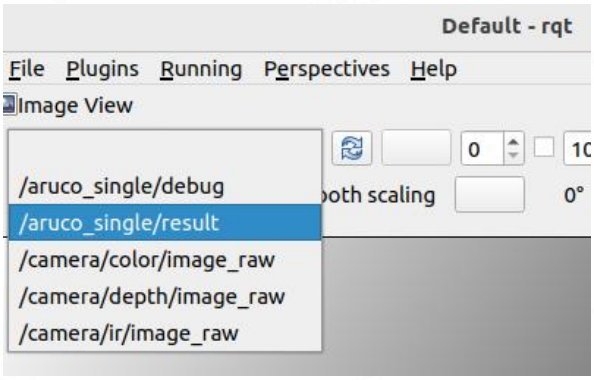
aruco.launch 文件可修改位置

2) 在登录后的终端输入启动 AR 标签识别功能的指令

```
roslaunch turn_on_wheeltec_robot aruco.launch
```

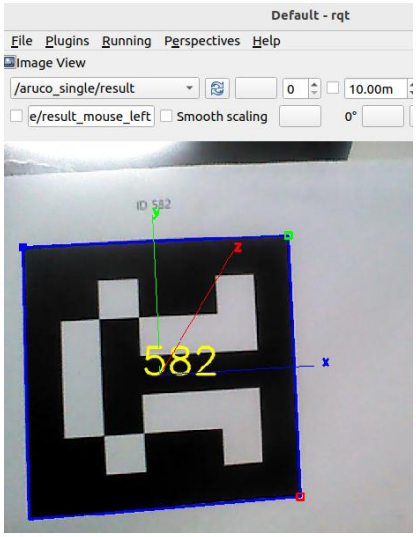
3） 功能启动完成初始化之后，我们可以通过 rqt 工具查看识别结果，在虚拟机端的终端运行 rqt_image_view。

选择话题/aruco_single/result



rqt 可视化选择对应话题

此时将我们前面生成的 id 为 582 的 AR 标签放置在小车相机视野内，可以看到标签已被正确识别，画面中将会显示 id 与坐标系



rqt 可视化查看 aruco 码识别效果

我们还可以通过查看话题的方式来获取所识别 AR 标签的位姿信息等，同样在虚拟机端的终端运行 rostopic echo /aruco_single/marker 指令查看 marker 话题。以下为一帧数据的内容节选，可以看到当前识别到的 AR 标签的位姿信息等

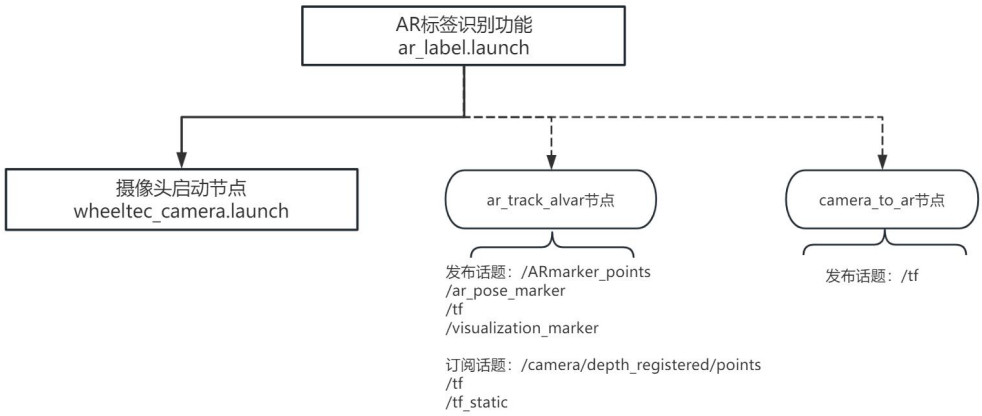
```
---
header:
  stamp:
    sec: 1679324571
    nanosec: 465318299
  frame_id: camera_link
ns: ''
id: 1
type: 1
action: 0
pose:
  position:
    x: 0.18187114596366882
    y: -0.1310642957687378
    z: 0.309150367975235
  orientation:
    x: 0.9996729469081757
    y: -0.020437675235274767
    z: 0.002410049415026072
    w: -0.015181973281441616
scale:
  x: 0.1
  y: 0.1
  z: 0.001
color:
  r: 1.0
  g: 0.0
```

topic echo 查看 aruco 码识别效果

4. 功能讲解

此处以 melodic 为例进行功能逻辑讲解。noetic 版本在 AR 标签识别功能上的实现逻辑与 melodic 版本一致，在启动 AR 相关节点的同时也启动相机节点，通过相机读取信息、aruco_ros 来处理读取到的信息，从而得到 AR 标签识别的效果。

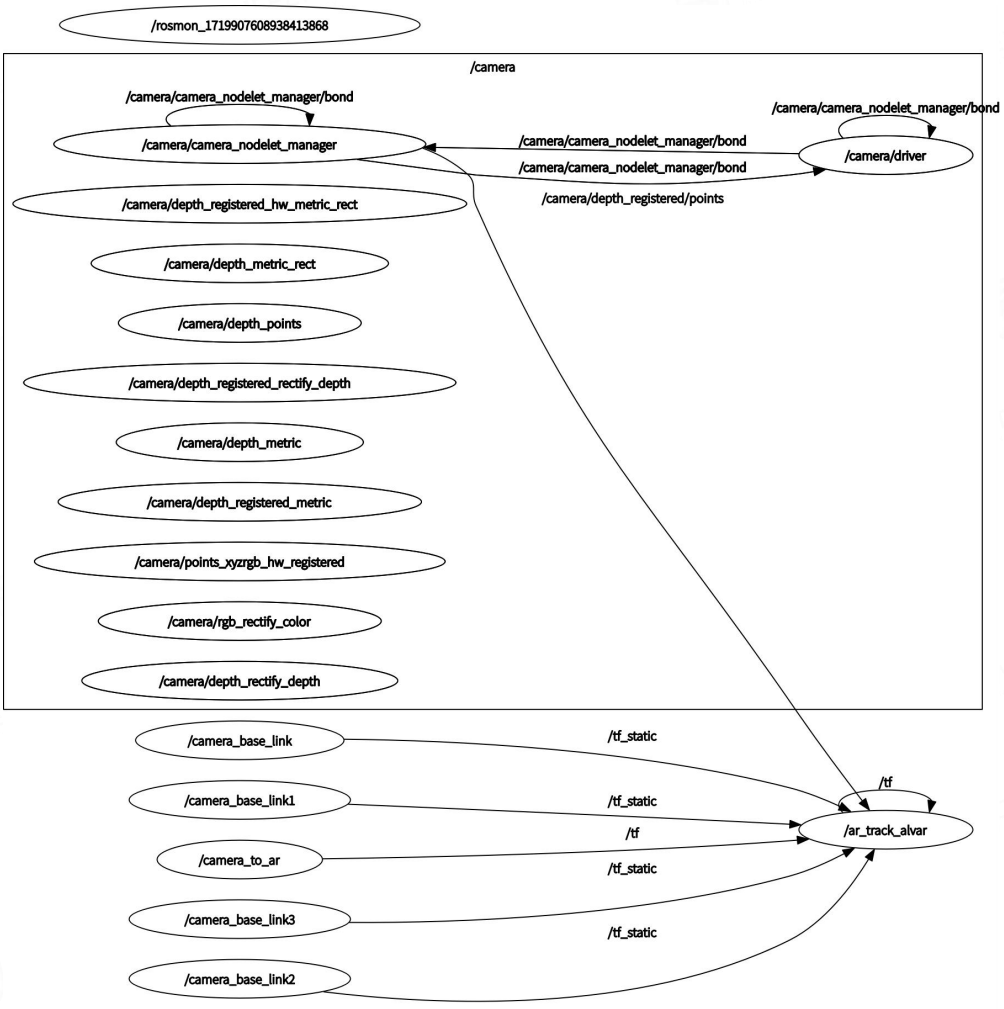
① 启动 AR 标签识别功能



AR 标签识别功能启动框架

② AR 标签识别功能节点关系

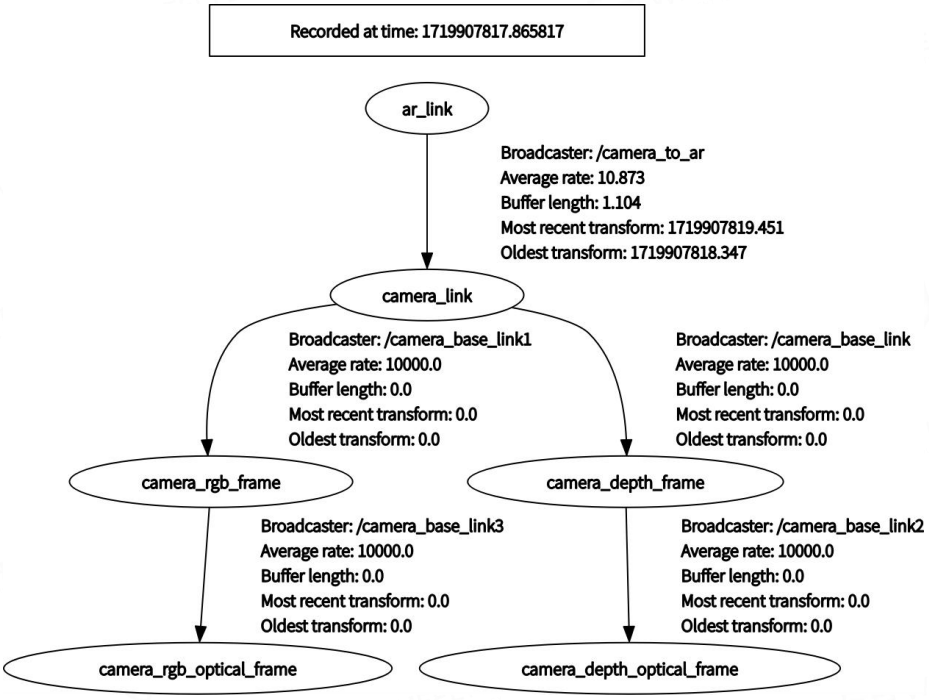
```
rqt_graph
```



AR 标签识别功能节点关系图

③ AR 标签识别功能 TF 树

```
rosrun rqt_tf_tree rqt_tf_tree
```



AR 标签识别功能 TF 树

④ AR 标签识别功能启动文件介绍

在我们的源码当中，与 AR 标签识别功能相关的只有一个文件，就是 ar_label.launch 文件，

ar_label.launch

```
<launch>
    <!-- 开启摄像头 -->
    <include file="$(find astra_camera)/launch/astrapro.launch" />
    <arg name="marker_size" default="4.4" />
    <arg name="max_new_marker_error" default="0.08" />
    <arg name="max_track_error" default="0.2" />

    <arg name="cam_image_topic" default="/camera/depth_registered/points" />
    <arg name="cam_info_topic" default="/camera/rgb/camera_info" />
    <arg name="output_frame" default="ar_link" />

    <!--连接camera_link和ar_link, 连接tf树 -->
    <node pkg="tf" type="static_transform_publisher" name="camera_to_ar" args="0 0 0 0 0 0  ar_link camera_link 100" />

    <node name="ar_track_alvar" pkg="ar_track_alvar" type="individualMarkers" respawn="false" output="screen">
        <param name="marker_size"           type="double" value="$(arg marker_size)" />
        <param name="max_new_marker_error"  type="double" value="$(arg max_new_marker_error)" />
        <param name="max_track_error"       type="double" value="$(arg max_track_error)" />
        <param name="output_frame"          type="string" value="$(arg output_frame)" />

        <remap from="camera_image"  to="$(arg cam_image_topic)" />
        <remap from="camera_info"   to="$(arg cam_info_topic)" />
    </node>

</launch>
```

在 ar_label.launch 文件中，设置了一些参数，我们可以通过查阅 ROS wiki 得到这些参数的含义：

- marker_size (double)——黑色方形标记边框一侧的厘米宽度，即 AR 标签的尺寸大小，ROS wiki 提供的 AR 标签默认边长大小为 4.4cm，因此

这里我们设置默认值 4.4，该参数可根据实际使用的 AR 标签尺寸进行更改，以达到更准确的识别效果；

- max_new_marker_error (double)——在不确定情况下何时可以检测到新标记的阈值；
- max_track_error (double)——决定在标记被认为已经消失之前可以观察到多少跟踪错误的阈值；
- camera_image(string)——AR 标签识别订阅的图片话题名称，该参数要求使用点云类型话题信息，这里我们使用深度点云话题；camera_info (string)——提供摄像机校准参数的话题名称，以便图像可以被校正，这里我们使用 RGB info 话题；
- output_frame (string)——AR 标签发布的笛卡尔坐标位置相对于的帧的名称，这里我们可以选择将它设定为 ar_link 或者直接使用 camera_link，如果在 output_frame 设置了未定义的坐标系，我们就需要在下面做一个 tf 坐标变换，因为实际坐标系为 camera_link。

整个 launch 文件的内容概括起来就是设置参数、打开摄像头、运行 ar_track_alvar 节点的过程，内容比较简单。