目录

蓝鲸智能机器人 ORB_SLAM2 包配置手册

1. 概述

ORB_SLAM2 属于基于稀疏特征点的视觉 slam 算法，蓝鲸智能机器人在原始版本的基础上进行了改进和优化。通过融合惯导里程计和激光雷达，提高了定位精度和定位鲁棒性，解决了单目尺度恢复和漂移问题，同时可以输出 2d 栅格地图。增加了地图保存、加载功能，发布相关 ROS 话题数据，使 ORB_SLAM2 兼容 ROS 的 navigation 框架。

蓝鲸智能机器人 ORB_SLAM2 包会输出完全兼容 gmapping 和 amcl 的 ros 接口数据，因此可以直接替换 gmapping 用于建图，替换 amcl 用于定位。

2. 可执行节点 nodes

共有 bwMono、bwMulti 两个 ROS node，使用方法是一样的。bwMono 可用于建图、定位、更新地图。bwMulti 目前只支持定位。

两个节点都需要传入两个参数，分别是字典文件和配置参数文件。字典文件相同，配置参数文件里面的参数选项是一样的，参数设置略有区别。下文将先介绍配置参数文件里面各个参数的意义，然后介绍两个节点特有的参数设置和 ros 话题信息。

对于节点的使用，蓝鲸机器人 ORB_SLAM2 包默认已经配置好了三个 launch 文件，map.launch 用于建图，run.launch 用于定位，update.launch 用于地图更新。用户可以根据下文教程和这些 launch 文件制作自己的 launch 文件。

Examples\ROS\ORB_SLAM2\Data 文件内的 show.rviz 文件可以用来在 rviz 中可视化建图过程。

2.1 节点参数

节点参数通过一个 yaml 文件传入，参数主要分成三个类别：orb 建图定位参数、栅格地图参数、传感器安装参数。Yaml 文件里面有很多参数是蓝鲸机器人开发测试用的，在本文没有提及如何修改的所有参数都需要保持默认值，否则可能导致错误结果或性能下降。

orb 建图定位参数：

~LoadMap，值为0或1，

0表示启动时不加载地图重新建图，1表示启动时加载地图。

~updateMap，值为0或1，

0表示关闭建图线程纯定位追踪模式，1表示开启建图线程用于建图和更新。

~drawTxt，值为0或1，

0表示关闭Frame图像里面的文字显示，1表示开启Frame图像里面的文字显示。

~LbaWeight，整数正值，默认值1000

用来调整融合惯导里程计数据在localmapping中的权重，值越小对应权重越小。默认值是推荐配置，不建议轻易修改；当惯导里程计飘逸比较严重时，降低这个值可以改善建图效果。

~GbaWeight，整数正值，默认值1

用来调整融合惯导里程计数据在全局图优化中的权重，值越小对应权重越小。默认值是推荐配置，不建议轻易修改；当闭环优化前的建图地图效果比较差时，提高这个值可以改善建图效果。

~dropRate，整数正值，默认值为2，

设置传感器数据丢弃间隔，值为2表示每2帧数据丢弃1帧。用来降低cpu运算量，实际应用不需要每帧数据都要处理。当机器人移动速度比较慢，传感器原始数据帧率比较高，cpu性能比较弱这些情形下，可以适当提高这个值。

~minInsertDistance，浮点数正值，默认值为0.3，

这个值表示地图中keyframe之间的最小距离，可以调整地图的稠密程度。当建图范围比较大时，适当的提高这个值可以极大的减小地图大小，提高建图效率和地图加载效率。当建图范围比较小时，适当的减小这个值可以增加地图的稠密度，提高定位的准确性和栅格地图的完整性。

~filterMap，值为0或1，

值为1表示地开启地图过滤功能，值为0表示关闭地图过滤功能。开启过滤功能后，地图保存时会先过滤地图，过滤主要是通过删除间隔小于（minInsertDistance* filerWeight）的keyframe来实现的。当大规模建图时，设置大的minInsertDistance虽然可以降低地图大小，但是会影响建图精度和稳定性，因此可以使用这个地图过滤功能。使用小的minInsertDistance值建立高质量地图后，保存时过滤成更精简的地图用于后续定位。

~filerWeight，浮点数正值，默认值为1，

地图过滤系数，用来调整保存地图的稀疏性。

~dbfile_rootpath，字符串，默认值slamdb

设置地图保存的根目录，即/home/xiaoqiang/$dbfile_rootpath为当前地图保存的根目录。

~map_name，字符串，默认值 default

设置地图保存的次级目录，即/home/xiaoqiang/$dbfile_rootpath /$map_name为当前地图保存的完整路径。

当设为default时表示为空，即/home/xiaoqiang/$dbfile_rootpath/为当前地图保存的完整路径。

~pub_mark，值为0或1，

值为1表示开启markers话题的发布，markers话题用于在rviz中可视化特征点和关键帧。值为0表示关闭markers话题的发布，用于节省系统资源。

栅格地图参数：

~border_minz，浮点数，默认值为-0.4，

特征点是三维空间中的点，用来投影生成栅格地图，只有map坐标系下z轴在[border_minz，border_maxz]范围内的点才参与。

Map坐标系中z轴等于零的平面在摄像头所在的水平面。

~border_maxz，浮点数，默认值为1.5，

特征点是三维空间中的点，用来投影生成栅格地图，只有map坐标系下z轴在[border_minz，border_maxz]范围内的点才参与。

~Odds_add，整数正值，默认值为30，

特征点或者雷达扫描点，生成栅格地图时增加障碍物的权重系数。增大该值可以让轮廓更明显，但是可能更厚更模糊。

~Odds_delet，整数正值，默认值为4，

特征点或者雷达扫描点，生成栅格地图时清除障碍物的权重系数。增大该值可以让轮廓边界更精细，但是也可能导致过滤太严重而不清晰。

~resolution，浮点数正值，默认值为0.05，

栅格地图的分辨率，单位为米

~pub_grid，值为0或1，

值为0表示不发布栅格地图（不影响保存地图时的栅格地图生成），值为1时表示每3秒发布一次当前栅格地图。建图时可以启用发布，用来实时观察建图进度和效果，定位时关闭以节省系统资源。

~use_scan，值为0或1，

值为1表示使用雷达扫描点来生成栅格地图，值为0表示使用特征点来生成栅格地图。没有雷达传感器时，需要设置为0。

传感器安装参数：

~ TbaMatrix，opencv矩阵，

摄像头坐标系到里程计base_link坐标系的转换矩阵，这是一个标准的4X4的旋转矩阵，它由摄像头安装参数决定，可以由3.2节得到。

对于bwMono节点，这个摄像头就是它使用的摄像头。对于bwMulti节点，这个矩阵对应1号摄像头。

~use_second，值为0或1，

值为1表示启用2号摄像头（用于bwMulti节点），值为0时表示关闭2号摄像头（用于bwMono节点）。

~ Tbc2Matrix，opencv矩阵，

Use_second参数值为1时才有效。和TbaMatrix一样，表示摄像头坐标系到里程计base_link坐标系的转换矩阵，只用于bwMulti节点，对应2号摄像头。通过把这个2号摄像头用于bwMono节点，使用3.2节标定教程得到TbaMatrix后，直接拷贝到Tbc2Matrix就可以使用。

~EnableCalib，值为0或1，

值为0表示关闭摄像头校准用于建图和定位，值为1表示开启摄像头标定功能。

~ TbsMatrix，opencv矩阵，

激光雷达scan坐标系到里程计base_link坐标系的转换矩阵，这是一个标准的4X4的旋转矩阵，它由激光雷达到base_link的tf关系计算得到。

2.2 bwMono 节点

bwMono 使用一个 rgb 摄像头融合惯导里程计和激光雷达，可以用于建图、定位和标定摄像头安装矩阵。

2.2.1 订阅的话题数据

/camera/image_raw （sensor_msgs/Image）
摄像头原始图像话题。

/camera/image_info （sensor_msgs/CameraInfo）
摄像头的标定参数话题。

/Odom （nav_msgs/Odometry）
待融合的惯导里程计话题。

/scan （sensor_msgs/LaserScan）
待融合的激光雷达数据话题，推荐传入经 laser_filters 包过滤之后的数据，这样输出的栅格地图质量会好很多。

/map_save （std_msgs/Bool）
用来触发地图保存线程，值为 true 表示开始保存地图，false 为无效值。

2.2.2 发布的话题数据

/ORB_SLAM/Camera （geometry_msgs/Pose）
当前追踪的摄像头的姿态，在 ORB_SLAM/World 坐标系下

/ORB_SLAM/Frame （sensor_msgs/Image）
当前追踪的摄像头特征点图像，用于可视化特征点在摄像头图像中的分布。

/ORB_SLAM/GBA （std_msgs/Bool）
值为 true 表示当前处于闭环优化过程中，此时应该暂停移动机器人，因为闭环优化过程中建图线程会被冻结。值为 false 表示不处于闭环优化过程，闭环优化已经结束。

/ORB_SLAM/GC （std_msgs/Bool）
值为 true 表示当前处于 gc 状态，值为 false 表示已退出 gc 状态。

/ORB_SLAM/Map (visualization_msgs/Marker)
用于在 rviz 中三维可视化特征点、关键帧等数据。

/ORB_SLAM/TrackingStatus (std_msgs/Int32)
用来指示当前视觉追踪状态，-1 表示系统没有启动完成，0 表示没有图像话题输入，1 表示视觉初始化还没有完成，2 表示视觉追踪正常，3 表示视觉处于丢失状态。

/bWmono/Odom (nav_msgs/Odometry)
视觉里程计，在 map 坐标系下，child frame_id 是 base_footprint。

/map (nav_msgs/OccupancyGrid)
当前建立的栅格地图

2.2.3 dynamic_reconfigure 参数

~update_map (bool)
设为 true 表示开启建图线程，用于建图、地图更新、摄像头标定。
设为 false 表示关闭建图线程，纯定位模式。

~enable_gba (bool)
请保持为默认的 true。

2.2.4 发布的 tf 变换关系

ORB_SLAM/World⟶ORB_SLAM/Camera
摄像头在 ORB_SLAM 的 world 坐标系下的当前姿态。

map⟶ORB_SLAM/World
ORB_SLAM 的 world 坐标系转到 map 坐标系的 tf 关系，这是一个静态变换，实际等价于参数里面的 TbaMatrix 矩阵

map→odom
机器人在 map 坐标系下的当前姿态，用于 ros 的 navigation 框架。

2.3 bwMulti 节点

bwMulti 是一个纯定位程序，定位用的地图是 bwMono 建立的。它使用两个摄像头融合惯导里程计进行定位，常用的配置是机器人头部和尾部各安装一个摄像头。它和 bwMono 的区别在于，由于使用两个摄像头进行定位，地图可以只建立一个视角方向的，运行时 bwMulti 会自动选择当前可以追踪的摄像头用于定位。

对于一个通常的闭合路径来说，因为摄像头的视野一般都小于 180 度，所以对于路径的任何一点，机器人建图时都需要看前后两个方向（顺时针建图一遍后再逆时针建图一遍），使用 bwMulti 则只用建立一个方向的地图。因此 bwMulti 可以极大的减轻建图工作量，提高机器人可移动范围。

2.3.1 订阅的话题数据

/camera1/image_raw （sensor_msgs/Image）
1 号摄像头原始图像话题。

/camera1/image_info （sensor_msgs/CameraInfo）
1 号摄像头的标定参数话题。

/camera2/image_raw （sensor_msgs/Image）
2 号摄像头原始图像话题。

/camera2/image_info （sensor_msgs/CameraInfo）
2 号摄像头的标定参数话题。

/Odom （nav_msgs/Odometry）
待融合的惯导里程计话题。

2.3.2 发布的话题数据

/ORB_SLAM/Camera （geometry_msgs/Pose）
当前追踪的摄像头的姿态，在 ORB_SLAM/World 坐标系下

/ORB_SLAM/Frame （sensor_msgs/Image）
当前追踪的摄像头特征点图像，用于可视化特征点在摄像头图像中的分布。

/ORB_SLAM/Map （visualization_msgs/Marker）
用于在 rviz 中三维可视化特征点、关键帧等数据。

/ORB_SLAM/TrackingStatus (std_msgs/Int32)

用来指示当前视觉追踪状态，-1 表示系统没有启动完成，0 表示没有图像话题输入，1 表示视觉初始化还没有完成，2 表示视觉追踪正常，3 表示视觉处于丢失状态。

/bWmono/Odom (nav_msgs/Odometry)

视觉里程计，在 map 坐标系下，child frame_id 是 base_footprint。

/map (nav_msgs/OccupancyGrid)

当前建立的栅格地图

2.3.3 dynamic_reconfigure 参数

~update_map (bool)

必须设为 false。

~enable_gba (bool)

必须设为 false。

2.3.4 发布的 tf 变换关系

ORB_SLAM/World⟶ORB_SLAM/Camera

摄像头在 ORB_SLAM 的 world 坐标系下的当前姿态。

map⟶ORB_SLAM/World

ORB_SLAM 的 world 坐标系转到 map 坐标系的 tf 关系，这是一个静态变换，实际等价于参数里面的 TbaMatrix 矩阵

map→odom

机器人在 map 坐标系下的当前姿态，用于 ros 的 navigation 框架。

3. 配置教程

单目建图的参考 launch 文件是 Examples\ROS\ORB_SLAM2\Data\map.launch。
地图更新的参考 launch 文件是 Examples\ROS\ORB_SLAM2\Data\update.launch。
单目定位的参考 launch 文件是 Examples\ROS\ORB_SLAM2\Data\run.launch。
多目定位的参考 launch 文件是 Examples\ROS\ORB_SLAM2\Data\run_multi.launch。

3.1 单目建图

使用的节点是 bwMono，在 launch 里面用 remap 配置好订阅的话题来源和参数配置文件，dynamic_reconfigure 的两个参数都需要配置为 true。

下文重点介绍参数配置文件里面必须配置的参数，其它参数请参考上文介绍自行调整。

~LoadMap，配置为 0 ，表示重新开始建立地图。

~updateMap，配置为 1 ，表示开启建图功能。

~EnableCalib，配置为 0 ，表示禁用摄像头标定功能。

~TbaMatrix，单目摄像头安装矩阵，由下节的标定教程获得，对于小强开发平台用户，出厂已经标定好。

~use_scan，配置为 0 ，表示不使用激光雷达。

配置完 launch 后，启动 launch 就可以开始建图。ORB_SLAM2 有一个视觉初始化的过程，需要直线移动机器人才能初始化，初始化成功时/ORB_SLAM/TrackingStatus 话题的值将由 1 变成 2。建图过程不能原地旋转，转弯时需要一边直线移动一边旋转。机器人可以倒退。建图过程中，如果视觉状态丢失了，需要遥控机器人回退到最近可以追踪的地方，ORB_SLAM2 的重定位功能会使视觉重新处于追踪状态，然后继续建图。

在关闭建图前，通过发布/map_save 话题可以保存地图用于后续的定位。

3.2 单目摄像头安装矩阵的标定

我们通常使用的是 usb 摄像头，它的 ros 驱动包是 usb_cam 包。bwMono 和 bwMulti 需要同时订阅摄像头图像和内参话题，因此 usb_cam 包的 launch 文件需要配置成同时发布摄像头图像和内参话题。摄像头内参的标定教程，请参考这篇教程：https://community.bwbot.org/topic/486/如何标定单目摄像头。

这里标定的是摄像头的安装矩阵，和摄像头内参标定不同，它属于外参标定。标定的原理是利用单目建图过程，通过求解视觉定位姿态和里程计姿态之间的最小二乘方程来得到摄像头的安装矩阵。

Launch 文件和 3.1 节的 launch 文件是一模一样的，唯一的区别在于下面两个参数的设置。

~EnableCalib，配置为 1 ，表示开启摄像头标定功能。

~TbaMatrix，单目摄像头安装矩阵，任意设为一个旋转矩阵即可，必须满足旋转矩阵正交性要求，推荐直接从参考 launch 文件中拷贝。

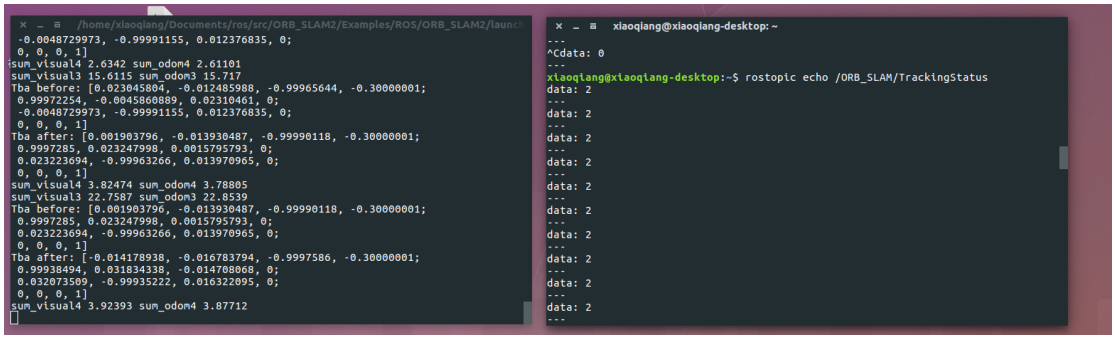
先根据机械安装参数，设置 TbaMatrix 的平移分量。假设 TbaMatrix 中 data 数据的成员分别为[R00, R01, R02, T0 ,R10, R11, R12, T1, R20, R21, R22, T2,0.0, 0.0, 0.0, 1.0],其中[T0, T1, T2]完全由机械安装位置决定，T0 表示摄像头在 base_link 坐标系下 x 方向的坐标值，T1 表示摄像头在 base_link 坐标系下 y 方向的坐标值，T2 设为 0。我们只用标定旋转分量[R00, R01, R02; R10, R11, R12; R20, R21, R22]。

整个标定分成两个阶段：1.初步标定，2.精确标定。整个标定过程，需要遥控机器人移动，推荐使用 vnc 远程登陆机器人配合遥控 app 进行操作。

初步标定：

启动 launch 文件后，手动遥控机器人直线前进，使 ORB_SLAM2 初始化成功，此时/ORB_SLAM/TrackingStatus 话题的值将由 1 变成 2。

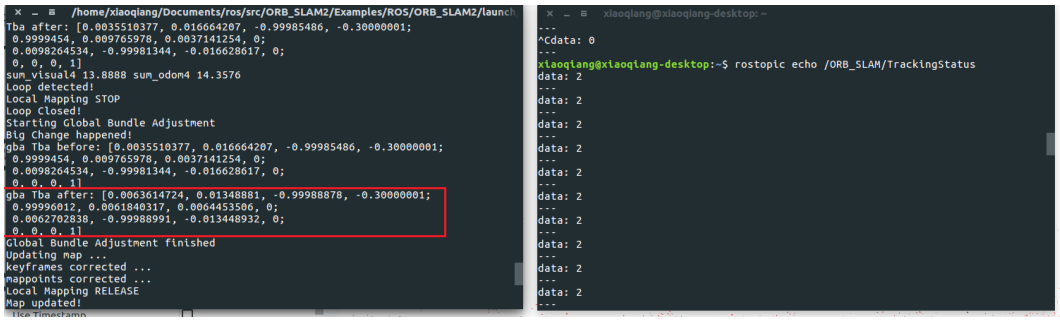
继续遥控机器人直线前进，同时观察 launch 文件运行窗口，会输出各种数据。其中 Tba after 就对应当前计算出的安装矩阵。当这个矩阵变化比较小，同时 sum_visual4 和 sum_odom 的比值接近 0.97 到 1.03 之间时，说明初步标定已经成功。



停止 launch 文件的运行，将当前计算得到的安装矩阵拷贝到 **TbaMatrix**。

精细标定:

在完成初步标定后，重新启动 launch 文件，然后遥控机器人移动完成一个闭合路径（通常是一个不小 3 米直径的圆），当触发全局闭环优化后，如下图 launch 运行窗口会输出一个更精确的安装矩阵，即 gba Tba after 后面的矩阵。



停止 launch 文件的运行，将当前计算得到的安装矩阵拷贝到 **TbaMatrix**。标定完成。

3.3 单目加 2d 激光雷达建图

使用的节点是 bwMono，launch 文件和 3.1 节是一模一样的，唯一区别在于下面两个参数的设置。

~use_scan，配置为 1 ，表示使用激光雷达。

~TbsMatrix，激光雷达 scan 坐标系到里程计 base_link 坐标系的转换矩阵，这是一个标准的 4X4 的旋转矩阵，它由激光雷达到 base_link 的 tf 关系计算得到（z 轴设为 0，roll、pitch 角也都设为 0）。

建图操作、注意事项和 3.1 节完全相同，保存地图后，不同的地方在于生成的栅格地图是由激光雷达点云生成的，地图轮廓会更清晰、漂亮。

3.4 单目地图更新

使用的节点是 bwMono，launch 文件和 3.1 节是一模一样的，唯一区别在于下面一个参数的设置。

~LoadMap，配置为 1 ，开启地图的加载功能，在上次地图基础上进行更新建图。

建图操作、注意事项和 3.1 节完全相同，推荐地图更新前先备份一下地图，以免更新失败导致原先的地图被破坏。结束更新前，需要保存地图才能保存更新。

3.5 单目导航定位

使用的节点是 bwMono，launch 文件和 3.1 节是一模一样的，区别在于下面两个参数的设置。

~LoadMap，配置为 1 ，开启地图的加载功能。

~updateMap，配置为 0 ，关闭地图的更新功能。

Launch 文件中 dynamic_reconfigure 的两个参数都需要配置为 false。

ORB_SLAM2 的视觉定位需要一个初始化过程，launch 文件启动后，需要让机器人视野处于建图范围内，此时才能通过重定位功能完成初始化。完成初始化后，/ORB_SLAM/TrackingStatus 话题的值将由 1 变成 2。

通过一个额外的简单 python 脚本可以完成 ORB_SLAM2 视觉定位的自动初始化，这个脚本不断控制机器人原地旋转的同时订阅/ORB_SLAM/TrackingStatus 话题，当侦测到/ORB_SLAM/TrackingStatus 话题的值由 1 变成 2 后，停止旋转机器人，此时 ORB_SLAM2 初始化完成，脚本可以退出运行。

3.6 前后两摄像头即双目导航定位

使用的节点是 bwMulti，launch 文件需要在 3.1 节基础上进行不少的修改，首先是摄像头话题数据需要重新映射。下面五个参数的设置也需要改正。

~LoadMap，配置为 1 ，开启地图的加载功能。

~updateMap，配置为 0 ，关闭地图的更新功能。

~use_second，配置为 1 ，同时启用两个摄像头进行定位。

~TbaMatrix，1 号摄像头安装矩阵，由 3.2 节教程得到。

~Tbc2Matrix，2 号摄像头安装矩阵，由 3.2 节教程得到，区别在于 T2 对应 z 轴，需要设置成 2 号摄像头相对 1 号摄像头的 z 轴投影距离，1 号摄像头所在水平面对应 z 轴为 0 的平面。

Launch 文件中 dynamic_reconfigure 的两个参数都需要配置为 false。

ORB_SLAM2 的视觉定位需要一个初始化过程，launch 文件启动后，需要让机器人视野处于建图范围内，此时才能通过重定位功能完成初始化。完成初始化后，/ORB_SLAM/TrackingStatus 话题的值将由 1 变成 2。

通过一个额外的简单 python 脚本可以完成 ORB_SLAM2 视觉定位的自动初始化，这个脚本不断控制机器人原地旋转的同时订阅/ORB_SLAM/TrackingStatus 话题，当侦测到/ORB_SLAM/TrackingStatus 话题的值由 1 变成 2 后，停止旋转机器人，此时 ORB_SLAM2 初始化完成，脚本可以退出运行。