Slam算法库使用说明

作者：赖贤城

邮箱：lxc.sysu@qq.com

时间：2019/08/07

简介

目的

本算法库的建立有两个目的：

1. **方便演示**，只需简单的几步操作就能演示算法运行
2. **方便使用**，需要用到某个算法的时候无需自己安装配置一遍，而可以用该算法库简单调整即可使用

包含的算法

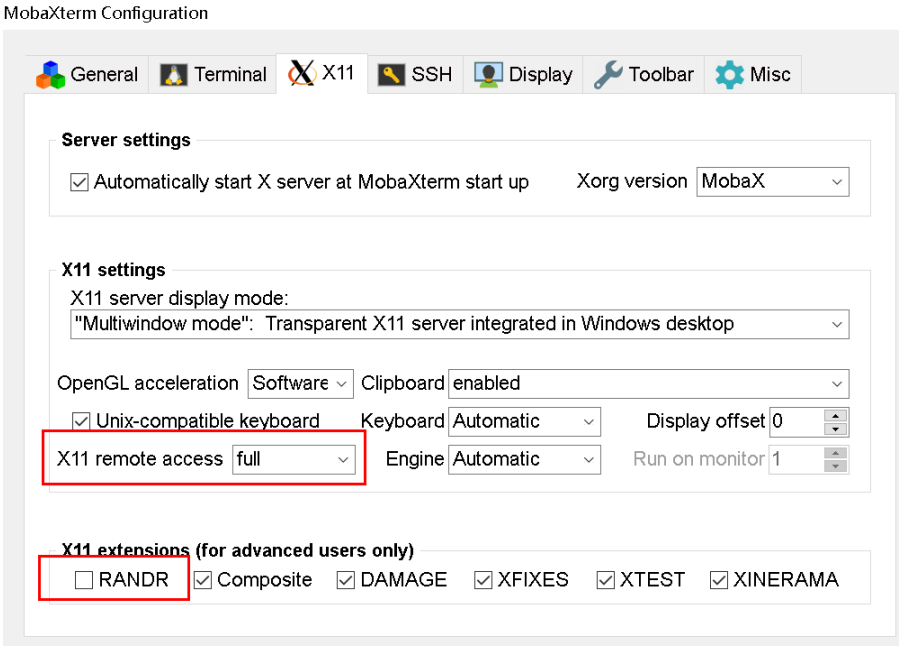
截至现在该算法库包含了10个算法，大部分为多传感器融合方法，具体为

1. `orb_slam2`，ORB-SLAM2是用于单目，双目和RGB-D相机的实时SLAM库，用于计算相机轨迹和稀疏3D重建，github地址：https://github.com/raulmur/ORB_SLAM2
2. `lsd_slam`，LSD-SLAM是一种实时单眼SLAM的方法。它是完全直接的（即不使用关键点/功能）。github地址：https://github.com/tum-vision/lsd_slam
3. `vins_mono`，VINS-Mono是单目视觉惯性系统的实时SLAM框架。它使用基于优化的滑动窗口配方来提供高精度的视觉惯性测距。github地址：https://github.com/HKUST-Aerial-Robotics/VINS-Mono
4. `dso`，DSO是一种新的直接和稀疏公式视觉里程。它结合了一个完全直接的概率模型（最小化光度误差）和所有模型参数的一致、联合优化。github：https://github.com/JakobEngel/dso
5. `svo`，一种半直接单眼视觉里程计算法，这种半直接方法不需要昂贵的特征提取和鲁棒匹配技术来进行运动估计。github地址：https://github.com/uzh-rpg/rpg_svo
6. `rovio`，基于EKF的鲁棒视觉惯性里程计，github地址：https://github.com/ethz-asl/rovio
7. `basalt`，具有非线性因子恢复的视觉 - 惯性映射，github地址：https://github.com/VladyslavUsenko/basalt-mirror
8. `msckf`，一种基于多状态约束卡尔曼滤波器的双目视觉里程计，github地址：https://github.com/KumarRobotics/msckf_vio

9. `okvis`，基于关键帧的视觉惯性里程计使用非线性优化，github地址：https://github.com/ethz-asl/okvis
10. `ice_ba`，对视觉惯性SLAM的一种增量式，一致且高效的光束平差法，github地址：https://github.com/baidu/ICE-BA

准备工作

1. 以下命令请使用MobaXterm运行，且将该软件的 Settings->X11中的改为如下



2. 输入命令`xclock`，若能看到一个时钟就没问题
3. 输入`xhost +`，解除Xserver的访问限制
4. 脚本使用了两个python包---`argparse`和`pyqt5`，如果你所在的环境没有这两个包请先安装它们
5. **在任意一台服务器运行**
    1. 下载镜像

```
docker pull freshorange/ros_slam_image:19-8-13
docker pull freshorange/ros_slam_image:kinetic_19-8-13
docker pull freshorange/no_ros_slam:basalt-v3
```

    2. 运行上述三个镜像，并将其中的`/root/setup_dataset.sh`中的数据集路径修改成该服务器能访问到的路径，之后保存为镜像，命令大概如下：

```
docker run -it --name ros_slam_14 freshorange/ros_slam_image:19-8-13
/bin/bash
(在容器中修改)vi /root/setup_dataset.sh
docker commit ros_slam_14 freshorange/ros_slam_image:19-8-13

docker run -it --name ros_slam_16 freshorange/ros_slam_image:kinetic_19-
8-13 /bin/bash
(在容器中修改)vi /root/setup_dataset.sh
docker commit ros_slam_16 freshorange/ros_slam_image:kinetic_19-8-13

docker run -it --name no_ros freshorange/no_ros_slam:basalt-v3 /bin/bash
(在容器中修改)vi /root/setup_dataset.sh
docker commit no_ros freshorange/no_ros_slam:basalt-v3
```

3. 下载外层封装代码

   `git clone https://github.com/Fresh-Orange/slam_lib.git && cd slam_lib`

图形界面使用

运行`python3 slam_gui.py`，将看到下图的图形界面，选择一种算法并选择一个数据集之后点击运行即可。当需要关闭所有运行窗口时，只需要点击“终止运行”即可全部关闭

命令行方式使用

运行`python3 slam_cmd.py --help`将看到命令行运行的使用方法。

该脚本有三个参数：

`--algo`：要运行的算法

`--dataset`：算法要运行在哪个数据集上

`--algo_dataset`：仅用于查看某个特定算法支持的数据集

使用举例：

1. 查看`orb_slam`支持的数据集

   `python slam_cmd.py --algo_dataset orb_slam`

   输出为`available datasets for orb_slam: ['tum', 'kitti', 'euroc']`

2. 运行LSD-slam

   `python3 slam_cmd.py --algo lsd_slam --dataset lsd_room`

   将会出现lsd-slam的运行界面

写给开发人员的使用方法

解开封装

上述脚本运行算法时存在3层封装，这里自上而下一层一层解开封装。

1. **python脚本层封装**

   目的：该层封装目的是简化不同算法不同数据集的调用方式，使得用户能够方便的选择想要运行的算法与数据集

   实现：

   - `slam_gui.py`使用`pyqt5`制作了简单的图形界面，用户点击选择算法(数据集)时记录下选择，点击运行时则调用下层封装
   - `slam_cmd.py`使用`argparse`实现了参数的传递，并通过一些判断逻辑确保用户的输入是可执行的，然后调用下层封装

   调用下层：

   主要是`slam_gui.py`和`slam_cmd.py`中的`docker_cmd`这个字符串，这个命令使用docker运行了一个特定的镜像并调用了其内部的脚本，以下述的命令为例，解析这个命令的含义

   ```
   docker run --env="DISPLAY"  \
                  --net=host  \
                  --volume="$HOME/.Xauthority:/root/.Xauthority:rw"  \
                  --env="QT_X11_NO_MITSHM=1" \
                  -v /tmp/.X11-unix:/tmp/.X11-unix:ro \
                  -it --rm --name slam_from_qt  \
                  -v /home:/out_home ros_slam_image:19-7-22  \
                   /bin/bash -c "/bin/bash /root/run_orb_slam.sh tum"
   ```

   `上述前五行的参数`：都是为了能在docker中正常显示图形界面而需要传递进去的参数

   `--name slam_from_qt`：赋予这个容器的名字，方便之后通过名字kill这个容器

   `-v /home:/out_home`：表示将本地的home目录挂载到docker容器中的out_home目录，这是为了在docker容器中能够访问到外部的数据集

   `ros_slam_image:19-7-22`：要运行的镜像，这里要运行的orb_slam就安装在这个镜像之中

   `/bin/bash -c "/bin/bash /root/run_orb_slam.sh tum"`：要在容器之中运行的命令，即在容器中运行`/bin/bash /root/run_orb_slam.sh tum`

因此，这个命令的执行结果就是“**从某个镜像运行一个容器，这个容器拥有运行图形界面的能力，并在这个容器中运行一行命令**”

2. **docker层封装**

   目的：为了方便算法库的转移。可以轻松的将算法库迁移到另一台电脑中运行，而不需要重新部署繁杂的环境

   实现：

   1. 以官方ros的`ros:indigo`为基础镜像，在此镜像上安装了`dso, svo, okvis, orb_slam, lsd_slam, rovio, ice_ba`等算法，安装结束后保存为`ros_slam_image:19-8-13`镜像
   2. 以官方ros的`ros:kinetic`为基础镜像，在此镜像上安装了`vins_mono, msckf`算法，安装结束后保存为`ros_slam_image:kinetic_19-8-13`镜像
   3. 以官方ubuntu的`ubuntu:18.04`为基础镜像，在此镜像上安装了`basalt`算法，安装结束后保存为`no_ros_slam:basalt-v1`镜像

3. **bash脚本层封装**

   目的：

   1. 为了整合运行命令。slam算法（特别是基于ros运行的算法）常常需要开启多个终端运行多个命令，这一层封装将这些命令打包，使得可以用统一的一句命令调用运行算法。
   2. 在此层实现了数据集选择功能

   实现：

   将所有的算法都整合成统一命名格式（run_XXX.sh）的脚本，并放在同一地方（/root目录），`ros_slam_image:19-7-22`镜像的root目录如下所示

```
root@Dog:/# ls root | grep run
run_dso.sh
run_ice_ba.sh
run_lsd_slam.sh
run_okvis.sh
run_orb_slam.sh
run_rovio.sh
run_svo.sh
```

   本层的封装根据不同算法的具体情况其命令大不相同，这里以orb_slam算法为例，run_orb_slam.sh的内容（仅仅显示部分）如下：

```
#TUM Dataset
if [ "$1" == "tum" ]; then
/root/ORB_SLAM2/Examples/Monocular/mono_tum /root/ORB_SLAM2/Vocabulary/ORBvoc.txt /root/ORB_SLAM2/Examples/Monocular/TUM3.yaml
/out_home/share_home/liul/project/slam/datasets/TUM/freiburg3_long_office_household
fi

#KITTI dateset
if [ "$1" == "kitti" ] && [ "$#" == "1" ]; then
/root/ORB_SLAM2/Examples/Monocular/mono_kitti /root/ORB_SLAM2/Vocabulary/ORBvoc.txt /root/ORB_SLAM2/Examples/Monocular/KITTI00-
02.yaml /out_home/share_home/liul/project/slam/datasets/KITTI/data_odometry_gray/sequences/01
fi
```

   上述内容显示该脚本中根据用户传入的参数选择不同的运行命令和数据集运行，并将冗长的命令封装在脚本内部。

进行修改

首先，你不必担心你的修改会破坏这个算法库，这就像我们使用ubuntu镜像进行各种安装配置但是并没有影响原来的ubuntu镜像一样。

当然，你如果希望你的修改能固化为镜像，那么请参考`docker commit`命令，将你的容器变为一个镜像。

这里**以orb_slam为例，假设你需要orb_slam运行在一个新的数据集上面**：

1. 开启docker容器

   orb_slam安装在镜像`ros_slam_image:19-7-22`中，因此我们运行此镜像得到一个容器(见下述命令)，其中的容器名称随意

   ```
   docker run --env="DISPLAY"  \
                  --net=host  \
                  --volume="$HOME/.Xauthority:/root/.Xauthority:rw"  \
                  --env="QT_X11_NO_MITSHM=1" \
                  -v /tmp/.X11-unix:/tmp/.X11-unix:ro \
                  -it --rm --name whatever_you_like  \
                  -v /home:/out_home ros_slam_image:19-7-22  \
                   /bin/bash
   ```

2. 准备好数据文件并修改脚本文件

   在本例中需要准备yaml文件和数据集，并修改/root/run_orb_slam.sh中的命令

   ```
   /root/ORB_SLAM2/Examples/Monocular/mono_tum /root/ORB_SLAM2/Vocabulary/ORBvoc.txt /root/ORB_SLAM2/Examples/Monocular/TUM3.yaml
   /out_home/share_home/liul/project/slam/datasets/TUM/freiburg3_long_office_household
   fi
   ```

3. 运行脚本

   使用`/bin/bash run_orb_slam.sh`运行脚本，这里使用bash而不是sh，因为有些脚本中运行了`source`命令，使用sh运行这些脚本会有问题